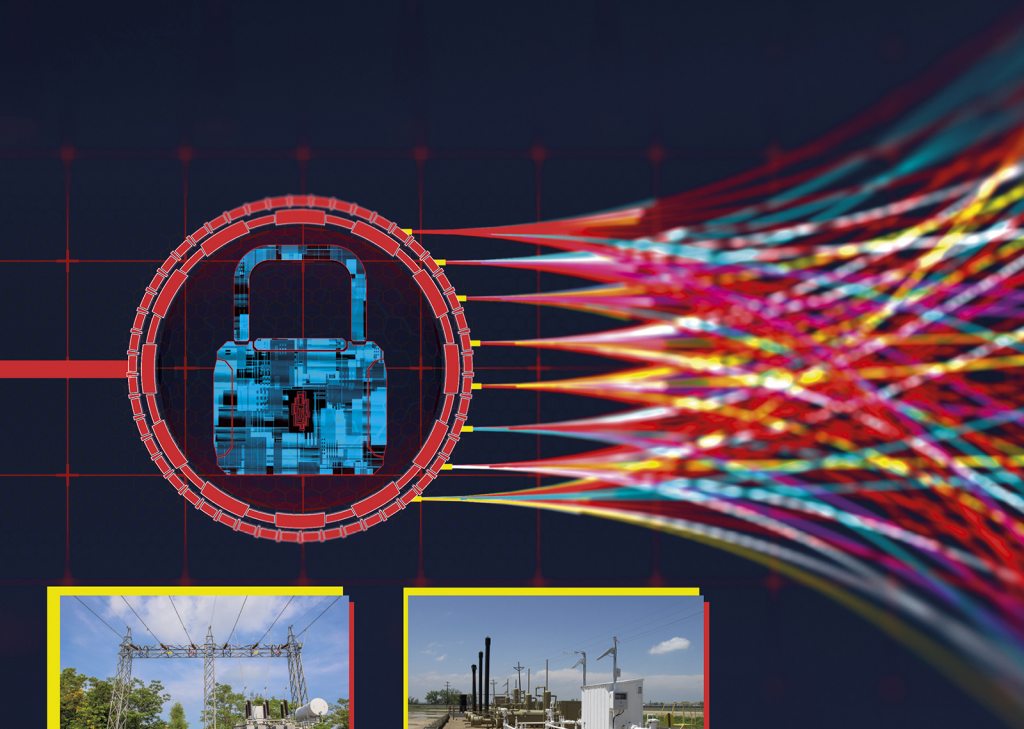
WILLIAM T. SHAW

CYBERSECURITY FOR SCADA SYSTEMS



2ND EDITION

Cybersecurity for SCADA Systems

CYBERSECURITY
FOR SCADA SYSTEMS

SECOND EDITION

PB
PENNWELL BOOKS

Disclaimer

The recommendations, advice, descriptions, and the methods in this book are presented solely for educational purposes. The author and publisher assume no liability whatsoever for any loss or damage that results from the use of any of the material in this book. Use of the material in this book is solely at the risk of the user.

PennWell Books, LLC
10050 E 52nd Street
Tulsa, OK 74146

866-777-1814
sales@pennwellbooks.com
www.pennwellbooks.com

Publisher: Matthew Dresher

Cover images: © iStock / Getty Images Plus - MF3d
© iStock / Getty Images Plus - TheYok
© Pipeline Knowledge, LLC

Library of Congress Cataloging-in-Publication Data

Names: Shaw, William T., author.
Title: Cybersecurity for industrial scada systems / William T. Shaw.
Description: Second edition. | Tulsa, OK, USA : PennWell Books, LLC, [2020] | Includes bibliographical references and index. | Summary: "Cybersecurity for SCADA Systems provides a high-level overview of SCADA technology, with an explanation of each market segment. Readers will understand the vital issues, and learn strategies for decreasing or eliminating system vulnerabilities"— Provided by publisher.
Identifiers: LCCN 2020044734 (print) | LCCN 2020044735 (ebook) | ISBN 9781593705060 (hardback) | ISBN 9781593705053 (epub)
Subjects: LCSH: Supervisory control systems. | Automatic data collection systems. | Data protection. | Computer security.
Classification: LCC TJ222 .S53 2020 (print) | LCC TJ222 (ebook) | DDC 620/.46028558—dc23
LC record available at https://lccn.loc.gov/2020044734
LC ebook record available at https://lccn.loc.gov/2020044735

1 2 3 4 5 23 22 21 20 19

Contents

Preface xvii
Acknowledgements xix
Introduction: Industrial Automation in the Aftermath of 9/11 xxi

Chapter 1

The technological evolution of scada systems 1
The Early History of SCADA—Mainframes 1
Minicomputers and Microprocessors 8
Central Architectures 10
Distributed Architectures 12
Client/Server Designs 14
Technological Convergence 15
Ubiquitous Internet and IP Networking 17
Generalized Software Architecture 20

Chapter 2

Remote terminal units 23
Basic Features and Functions 23
Smart RTU Technology 29
Top-Down and Bottom-Up Configuration 47
The Emergence of PLCs 49
Legacy Protocols 50
Protocol Standards 54
IP-Ready RTUs and Protocols 57

Chapter 3

Telecommunications technologies 61
Voice-Grade (Analog) Telephony 61
Commercial Voice/Data Carriers 69
Options for Wireless Communications 76
Digital Networking Technologies 80
TCP/IP Networking 84
The Internet 94

Chapter 4

Supervisory control applications 99
Operating System Utilities 100
SCADA System Utilities 105
Program Development Tools 115
Standardized APIs 121

Chapter 5

Operator interface 129
Access-Control Mechanisms 129
Standard System Displays 132
Site/Industry–Specific Displays 138
Historical Trending 156
Logs and Reports 164

Chapter 6

Conventional information technology 169
Availability, Integrity, and Confidentiality 170
Remote Access/Connectivity 172
TCP/IP Suite 177
Firewalls & Routers 184
Wireless LANs 206
Authentication and Validation 208
Encryption and Ciphers 214

Chapter 7

Identifying cybersecurity vulnerabilities 229
Threats and Threat Agents 229
Obvious Points of Attack and Vulnerability 238

Chapter 8

Malware, cyberattacks and hacking tools 251
Vulnerabilities 268
WEB Server/SQL Injection 271
Email and Web browsing 273
Malware 275

Chapter 9

Physical security 279
Access Controls 280
Access tracking 287
Illegal-entry Alarms 288
Physical Isolation of Assets: Layers of Defense 289

Physical Protection of Materials and Information 289
Critical Ancillary Subsystems 291
Remote and Field Sites 294

Chapter 10

Operational security 297
Policies and Administrative Controls 297
Procedures 300
Operational Differences 304
Training 306
Recovery Procedures 307
Annual Review 308
Background Checks 309

Chapter 11

Computer systems & Network security 311

Chapter 12

Electric utility industry–specific cybersecurity issues 363
Substation Backdoors 366
IP to the Substation 370
TASE.2/ICCP Connections 372
UCA2 (IEC61850) 373
DNP3.0 374
NERC 1200/1300 Compliance 374

Chapter 13

Water/Wastewater industry–specific cybersecurity issues 377
Licensed Radio Communications 379
Nonsecure Protocols 381
PLC Equipment as RTUs 382
Supervisory and Local Control Applications 383
Municipal LANs and WANs 386
Control Interfaces to Plant Control Systems 387

Chapter 14

Pipeline industry–specific cybersecurity issues 389
Radio Communications 391
Smart RTUs 393
RTU Program Logic 394
Supervisory Control Applications 394
IP along the Pipeline 395
Web Browsing and Email Integration 396

Chapter 15
The cyberthreat to scada systems 397

Chapter 16
Commercial product vulnerabilities 403

Appendix A
U.S. Department of Energy's "21 Steps to Improved SCADA Security" 409

Appendix B
NERC CIP—Recommendations for Electric Utilities 415

Appendix C
Security Recommendations of the Instruments, Systems, and Automation Society and the American Gas Association 421
Recommendations of the AGA 423

Appendix D
Industry and Government Security Recommendations 425

Appendix E
SCADA System Security Assessment Checklists 427

Glossary 443

Index 477

Figures

Fig. 1–1. Simplified component diagram of a SCADA system 2
Fig. 1–2. Example bit-oriented message format (starting and ending portions only, owing to actual large number of bits required in a full-length message) 4
Fig. 1–3. Example tabular operator display 7
Fig. 1–4. Example semi-graphic operator display 7
Fig. 1–5. Centralized, redundant SCADA system architecture 11
Fig. 1–6. Distributed SCADA system architecture 13
Fig. 1–7. Client/server SCADA system architecture 15
Fig. 1–8. Virtualized SCADA system implementations 16
Fig. 1–9. SCADA as a service 18
Fig. 1–10. SCADA without a SCADA system 19
Fig. 1–11. Generalized Information Flow within a Generic SCADA System 21
Fig. 2–1. Typical MTU console 24
Fig. 2–2. RTU contact output (control) types 29
Fig. 2–3. Evolution of smart RTU technology and capabilities 30
Fig. 2–4. RTU hierarchy using master and slave protocol combination 33
Fig. 2–5. Typical RTU multiline LCD and keypad 35
Fig. 2–6. Example host definition of downloaded calculation functions 38
Fig. 2–7. Supervisory control of local regulatory control panels 39
Fig. 2–8. Basic SCADA system and DCS architectures 40
Fig. 2–9. IEC 61131 PLC/RTU configuration alternatives 42
Fig. 2–10. Categories of typical RTU protocol message types 51
Fig. 2–11. Simple RTU serial protocol architecture 55
Fig. 2–12. Network-based serial protocol architecture 56
Fig. 2–13. Ethernet cable 60
Fig. 3–1. SCADA host with multiple master radios on separate frequencies 66
Fig. 3–2. Typical microwave-based private telephone system 68
Fig. 3–3. Use of packet switching networks for SCADA communications 71
Fig. 3–4. Connection-oriented telephone circuits 72
Fig. 3–5. Using digital telephone circuits to bridge LANs 73
Fig. 3–6. Frame relay and FRADs used to replace analog leased lines 75
Fig. 3–7. Spectral energy (frequency) distribution of spread-spectrum radio . . . 77
Fig. 3–8. Cellular data communications architecture 79
Fig. 3–9. Frame-relay DLCI-to-IP-address mapping in routers 81
Fig. 3–10. FDDI counter-rotating ring design 83

Fig. 3–11. Typical corporate IP network architecture 86
Fig. 3–12. Some of the basic protocols in the IP suite 89
Fig. 3–13. Site-to-site and remote-access VPNs 92
Fig. 4–1. Software layers comprising a typical SCADA system host 100
Fig. 4–2. Evolution of SCADA software with commercial software 101
Fig. 4–3. SCADA system user account management utility 107
Fig. 4–4. SCADA configuration utilities 110
Fig. 4–5. Database point and calculated point creation utility 111
Fig. 4–6. Creating the tag database using a spreadsheet utility. 112
Fig. 4–7. Graphical display editor 114
Fig. 4–8. Application program interacting with HMI via SCADA library functions 117
Fig. 4–9. OPC client/server architecture and data-exchange alternatives 122
Fig. 4–10. Using a SQL-compliant database server to exchange SCADA data 124
Fig. 5–1. Example SCADA system control room console design 130
Fig. 5–2. Typical RTU polling and communications diagnostic display. 133
Fig. 5–3. SNMP-based RTU polling and communications diagnostic display 134
Fig. 5–4. SCADA system operational status display 135
Fig. 5–5. RTU current value display 137
Fig. 5–6. Point group display (bar graph mode). 138
Fig. 5–7. Web-page operational display 141
Fig. 5–8. Geographic layout operational display 142
Fig. 5–9. Process-flow operational graphical display 143
Fig. 5–10. Display hierarchy and inter-display navigation 145
Fig. 5–11. GIS example SCADA display 146
Fig. 5–12. Alarm limit checking on a typical analog input point 149
Fig. 5–13. Typical current-alarm summary display window. 151
Fig. 5–14. Using symbols or code letters to indicate measurement conditions 154
Fig. 5–15. Control-point tagging display. 156
Fig. 5–16. Mechanical strip-chart pen recorder. 157
Fig. 5–17. Data storage hierarchy for historical trending. 159
Fig. 5–18. Example historical trending display. 162
Fig. 5–19. A typical SCADA system event log query 165
Fig. 5–20. Example of a spreadsheet report for a water utility 167
Fig. 5–21. Microsoft Windows Task Scheduler Utility. 168
Fig. 6–1. Communication interconnections to a SCADA system 173
Fig. 6–2. Attacking and utilizing a legacy serial communication circuit 175

Fig. 6–3. Attacking a SCADA system that is using IP-to-the-Field 176
Fig. 6–4. The OSI seven-layer model and IP equivalent-function layers 179
Fig. 6–5. Performing NAT in a gateway computer 183
Fig. 6–6. IP and TCP (or UDP) datagram header information 185
Fig. 6–7. IP datagram fragmentation 187
Fig. 6–8. The internal structure of the IPv4 datagram header. 187
Fig. 6–9. The Zenmap network scanning tool 190
Fig. 6–10. IP routing between PC network interfaces 192
Fig. 6–11. Accidental bridging of LANs via dual-home connections 193
Fig. 6–12. Ethernet frame structure and contents 195
Fig. 6–13. Switched Ethernet LAN elements 197
Fig. 6–14. Creating a broadcast storm in an Ethernet LAN 199
Fig. 6–15. Using RSTP to re-establish LAN communications 199
Fig. 6–16. Tagged frames with different priority values 201
Fig. 6–17. Tagged and un-tagged Ethernet frames 202
Fig. 6–18. Setting up a SPAN port on a switch. 203
Fig. 6–19. IEEE 802.1x port-based NAC 205
Fig. 6–20. Using Syslog protocol to send logs to a SIEM 205
Fig. 6–21. Using Wi-Fi at SCADA field sites for local communications 207
Fig. 6–22. Cyber security measures if Wi-Fi must be used 208
Fig. 6–23. Remote and local user access to a computer/system 209
Fig. 6–24. Strong authentication options and technologies 211
Fig. 6–25. Encryption and decryption of a document 215
Fig. 6–26. Using stream cipher to protect transmitted information 216
Fig. 6–27. Key size increases time required to break 217
Fig. 6–28. Public and private key generation and exchange 217
Fig. 6–29. Public-private key encryption. 218
Fig. 6–30. MS Windows Encrypted File System 220
Fig. 6–31. Hash algorithm generating a message digest value 221
Fig. 6–32. Example of a public key (Base 64 encoded) 222
Fig. 6–33. Example of an X.509 digital certificate 223
Fig. 6–34. Creating zones and network segmentation with firewalls 226
Fig. 6–35. Conceptual design of a media scanning 'kiosk' 227
Fig. 7–1. Taxonomy of potential threat sources to a SCADA system 229
Fig. 7–2. Example of a spear phishing email 235
Fig. 7–3. Typical attack pathways for cyberattack 239
Fig. 7–4. Classes of security and types of countermeasures 240

Fig. 7–5. US-CERT/CISA monthly vulnerability updates 245
Fig. 7–6. The MITRE Corporation's ATT&CK™ Database 246
Fig. 7–7. The MITRE Corporation's IACS ATT&CK™ database 247
Fig. 8–1. The Metasploit framework with Armitage 254
Fig. 8–2. Sample exploits from the Metasploit framework 255
Fig. 8–3. Using Metasploit to inject a VNC into a target computer 256
Fig. 8–4. Kali Linux distribution with pen-testing tools 257
Fig. 8–5. Buffer overflow attack 269
Fig. 8–6. Stack smashing in an x86 CPU 271
Fig. 8–7. Relational database tampering via SQL injection 272
Fig. 9–1. Gates and fencing to control vehicle and personnel access 281
Fig. 9–2. Physical security layers for added security 282
Fig. 9–3. Various forms of high-security key/lock systems 283
Fig. 9–4. Typical electronic access-control door 285
Fig. 9–5. Electronic, automated access-control and intrusion-detection system 286
Fig. 9–6. Tamper-indicating/detection mechanisms 287
Fig. 9–7. Physical protection of roof areas with radio equipment 292
Fig. 9–8. Physical protection of network cabling and components 293
Fig. 9–9. Power supply: typical configuration and equipment interconnections 294
Fig. 11–1. Unified Threat Management appliance 314
Fig. 11–2. Application proxy firewalls 315
Fig. 11–3. Packet-inspection firewall 317
Fig. 11–4. Tracking TCP state for each session 317
Fig. 11–5. Ethernet switch with packet filtering 318
Fig. 11–6. Transparent firewall operation 321
Fig. 11–7. Industrial protocol-aware firewall 323
Fig. 11–8. Industrial protocol detailed filtering 324
Fig. 11–9. Cross-network vulnerabilities with IP to the field 325
Fig. 11–10. NIDS components and structure 326
Fig. 11–11. Using network taps for message collection 328
Fig. 11–12. Network intrusion detection and prevention system 329
Fig. 11–13. Host-based intrusion detection 332
Fig. 11–14. SNMP Client and Agent communications 334
Fig. 11–15. SNMP client software (network monitor) from SolarWinds™ 336
Fig. 11–16. Simplified block-diagram of a SIEM 339
Fig. 11–17. SNTP server on the LAN, with GPS time source 342

Fig. 11–18. Using a data diode to forward data safely 344
Fig. 11–19. Establishing a DMZ to isolate the SCADA system 345
Fig. 11–20. Setting port security on a Cisco switch port 346
Fig. 11–21. Using IEEE 802.1x port-based NAC 348
Fig. 11–22. Using Syslog messages to monitor network elements 349
Fig. 11–23. Blocking unnecessary ports in Windows 352
Fig. 11–24. Windows local security policy setting groups 353
Fig. 11–25. Windows security policy templates 354
Fig. 11–26. Possible incorrect placement of NIDS sensor when VPN is used 356
Fig. 11–27. Accessing VLANs using pre-tagged frames 357
Fig. 11–28. The "hosts" and "lmhosts" files in Windows 359
Fig. 11–29. Using a KVM to support multi-computer access 361
Fig. 12–1. Generalized block diagram of an electric utility SCADA system 365
Fig. 12–2. Electrical transmission substation circa 1990 367
Fig. 12–3. Substation information consolidation (substation automation) 370
Fig. 12–4. IP networking to the substation 371
Fig. 13–1. SCADA system block diagram for the water/wastewater industry 378
Fig. 13–2. Using portable equipment to hijack a remote site 381
Fig. 13–3. Evolution of PLC programming and configuration downloading . . . 383
Fig. 13–4. Using serial link cryptographic transceivers 385
Fig. 13–5. Example MAN shared by a municipal utility SCADA system. 387
Fig. 14–1. Generalized architecture of a pipeline SCADA system 390
Fig. 14–2. Evolution of pipeline communications technologies 392
Fig. 15–1. Cyber event categorizations 399
Fig. 16–1. The CVE database Web site based on MITRE data 404
Fig. 16–2. CVE vulnerability listing for Windows XP. 405
Fig. 16–3. Microsoft security bulletin Web site—example bulletin. 405
Fig. 16–4. The CVE Details Web page for Enterprise Linux vulnerabilities 406
Fig. 16–5. NIST National Vulnerability Database (NVD) 407
Fig. 16–6. Cisco's product support site for CVE assessment & support 408

Tables

Table 1–1. Trends in the evolution of system architecture 16
Table 4–1. Standard operating system administrative utilities 103
Table 4–2. SCADA configuration maintenance and management utilities 107
Table 4–3. Typical SCADA application programming library functions 119
Table 6–1. Ethertype values for some notable protocols 196
Table 7–1. Example social engineering techniques used by attackers 233
Table 7–2. Most troublesome vulnerabilities in the past decade 242

Preface

In the 1960s, when the first computer-based supervisory control and data acquisition systems (SCADA) were being developed, there was no cultural concept of needing to provide any particular protective measures to keep such systems safe from intentional attacks. After all, why would someone want to disrupt the operation of such systems? The world was a different place, and the computer expertise to work on, or with, such systems was a rare commodity. The only protective considerations built into those systems were ones instituted in order to minimize or eliminate the impact of user errors. Not so today. Computers have become commodity appliances, and computer expertise far more commonplace. In addition, there are people that have technical expertise and, for a variety of reasons, choose to use it to inflict damage. Or worse still, there are those who wish to use such expertise to cause serious harm to the United States government and citizens. The Internet, a world-spanning communications technology that should be a positive force to unite cultures and peoples, is also being used as a means to reach into our computer systems by such people. Much of our critical industrial infrastructure is managed and controlled by SCADA systems, and thus it is now essential that we place protective measures into and around these systems. This book is intended to provide a general background of SCADA system technology, cybersecurity concepts and technologies, and how the two can be brought together to safeguard our infrastructure and computer automation systems. In this second revision of the book, I have included a great deal more information about implementing cybersecurity protections and about technical countermeasures. This revision also takes advantage of the evolved industry-specific cybersecurity standards that have emerged since the initial printing, especially in the electric power and oil-and-gas pipeline industry sectors. There have also been many technological changes in communications and networking and other areas of computer science since the original publication. I have tried to capture applicable changes in this revision.

Acknowledgements

The author would like to acknowledge all of the people that assisted in the writing of this book and, in particular, my wonderful and understanding wife, who put up with the long nights and weekends spent writing, editing, and proofing this manuscript. I would also like to thank the people at PennWell, who encouraged me to update this book and who helped in editing it into a suitable, professional-looking document.

Introduction

Industrial Automation in the Aftermath of 9/11

Without the events of September 11, 2001, there might not have been a need for this book, or at least for such a book this soon. Up until the events of that day, the people and government of the United States held the belief that we were isolated and insulated from the foreign governments and "true believers" that might wish us harm. It is true that for many years we had been witnessing a growing problem of computer *hacking* and the regular introduction of computer *worms*, *viruses*, and other forms of *malware* over the Internet. But these activities were not perceived as serious, intentional attacks on our country or infrastructure. After 9/11, everything changed. We now know that there are people and groups that will spend the time and money to create havoc and terror in order to advance their political, social, or religious agendas. In response to the events of 9/11, the Department of Homeland Security (DHS) was formed and given the responsibility for protecting us from these people and organizations. One of the results of the work being done by the DHS was the recognition that the vast majority of our industrial and manufacturing facilities, technological infrastructure, transportation systems, and energy infrastructure are run and controlled by computer-based systems—and that these systems (mainly *SCADA* and *DCS/PLC* systems) were not designed with any intrinsic protective mechanisms. This is not to say that such systems were/are fragile or even readily accessible to an attacker. The vendors of these systems have generally designed them to be robust and capable of continued operation even with some level of component failures or damage. This was essential because of the critical or essential nature of the processes being controlled by such systems.

Designers of computer-based automation and control systems have always known that computers, and electronic devices in general, can and will fail. Thus, system designs have long accounted for this possibility through redundancy schemes and architectures that permitted "graceful degradation." In the early years of computer-based automation systems, these systems were typically employed in a "stand alone" configuration without any communications with, or interfaces to, other computer systems. The only way that a remote cyberattacker/hacker could access such systems would have been if a dial-in telephone circuit had been supplied with the system, for the purpose of providing remote support by the system vendor. But, as computer and networking technology became pervasive and ubiquitous in all aspects of modern business enterprises, these automation systems started to be interfaced with corporate networks, business systems, and eventually, even to the Internet itself. This evolutionary process has provided cyberattackers with much greater access to these critical automation systems.

Since this book's initial publication, we have become aware that hostile nation-states are actively supporting and funding hacking activities in order to steal our intellectual property and trade secrets, disrupt our elections, gain access to our bank accounts and credit cards, and generally establish remote access into business and governmental computer systems (including those of the military). Since the initial publication of this book, there have been numerous documented instances, around the globe, of cyberattacks on power grids, industrial facilities, water distribution systems, and communication systems. So, providing adequate and effective cybersecurity for our SCADA systems is even more important than ever.

The world of computer-based automation systems can be divided into two broad classes: those systems used with processes that are spread over a large geographical area (and thus require the use of wide-area communications technology) and those systems that manage processes that are geographically constrained (and thus can use local-area communications technology). The first type of system is called a Supervisory Control And Data Acquisition system (SCADA) and is used in applications such as gas and liquid pipelines, electric power transmission and distribution, and water distribution systems. The second type of system is called a Distributed Control System (DCS) and is used in plant automation applications such as refining, steel production, paper and pulp, food and beverage, bulk and fine chemicals, etc. A variation of this second type of system is based on Programmable Logic Controller (PLC) technology. Almost every high-volume manufacturing facility is automated using PLC control system technology. The DHS has initially focused its efforts on cybersecurity for SCADA systems, and therefore SCADA system cybersecurity will be the focus of this book, although many of the issues and principles will be directly relevant and applicable to DCS systems as well.

1

The technological evolution of scada systems

The Early History of SCADA—Mainframes

Supervisory control and data acquisition (SCADA) systems are used to monitor and remotely control critical industrial processes, such as gas pipelines, electric power transmission, and potable water distribution/delivery. As such, SCADA systems are important to our daily lives, even though most people never see them or even know of their existence.

To properly understand what SCADA systems are, how they came to be, and why they are designed the way they are, one needs a basic understanding of the history of SCADA system development. It is also helpful to know why things have evolved and what factors have pushed this evolution. Computer-based supervisory control systems were introduced in the 1960s, and the first such systems were based on the mainframe computer technology available at the time. These systems were not yet called SCADA systems, as that particular acronym did not come into general use until the 1980s. SCADA systems were developed to replace older technologies (e.g., tone telemetry) and to provide features and functions that required computational and logical capabilities. The incorporation of a computer into telemetry systems provided a means for manipulating, processing, storing, and presenting data that could not be provided with previous technologies.

In the late 1960s, the integrated circuit had been invented, making it possible to build complex and sophisticated (for the time) electronic devices, including mainframe computers. By their nature and purpose, SCADA systems are intended to provide a human *operator* with updating *real-time* information about the current state of the remote process being monitored, as well as the ability to manipulate the process remotely. There were four basic major components in a 1960s SCADA system (fig. 1–1): the usually fully redundant central computer (also called the *host* computer or master), the field-based remote measurement and control equipment (called *remote terminal units* [RTUs]), the wide-area telecommunications system to connect them, and the operator interface that provided user/operator access to the system (also called the *operator console,* human-machine interface [MMI], and

later, in a more politically correct age, human-machine interface [HMI]). Although computer and communications technologies have advanced since the 1960s, even modern SCADA systems have a similar architectural basis.

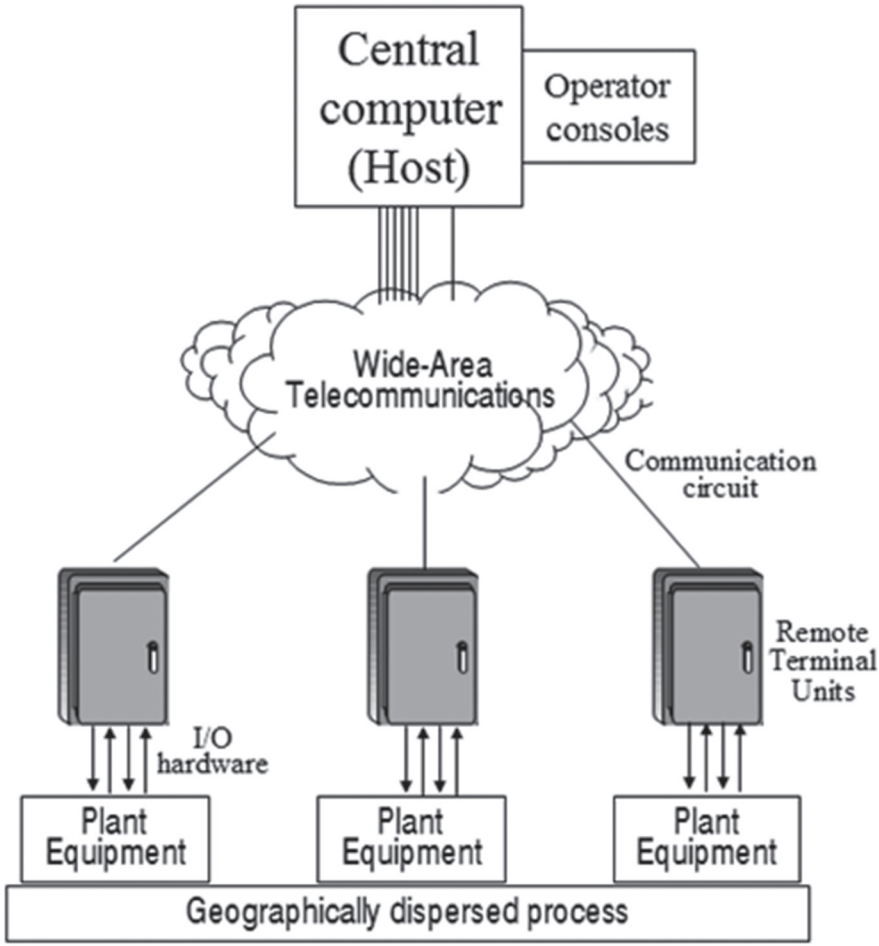


Fig. 1–1. Simplified component diagram of a SCADA system

To provide real-time data updates from the field, a SCADA system needs remote sensory and communications capabilities. Electronic devices called RTUs are located at each point where measurements are to be taken or where process equipment is to be controlled. The central computer continuously, cyclically polls the field-based RTUs to fetch their current measurement values and status indications. Polling is the process of sending a message to an RTU, to elicit a response message containing updated values, and repeating that operation with subsequent

RTUs, until all have been processed. Then, that sequence is repeated over and over, endlessly, presumably providing the SCADA operational personnel with a near-real-time view of the process being supervised and controlled. In the earliest SCADA systems, RTUs were essentially nothing more than remotely located I/O equipment that could read and transmit inputs and receive commands to generate outputs. That remained the case in the electric power transmission industry well into the 1990s. Other industry segments made strides in enhancing and utilizing the intelligence and capabilities of their RTU devices starting in the 1970s.

The RTUs designed in the 1960s, some of which remain in use today, were very simple electronic devices with a severely limited set of hardwired functions. They were 'digital' from the standpoint of being built from discrete logic gates (and, or, xor, nor, latch, not, flip-flops, etc.) but they contained no microprocessor or stored program logic. Normally, RTUs are equipped with input/output (I/O) hardware circuitry that enables the measurement of electrical signals generated by devices that produce voltages or currents in proportion to a physical process parameter such as pressure, flow rate, level, or temperature. (The term *transmitter* is often used for the device that produces an electrical signal proportional to the value of a physical property.) RTUs are also able to generate control outputs, either as voltage or current signals, or in the form of switch/contact closures. Output control signals are initiated or adjusted by the RTU upon receipt of a command from the central computer, usually initiated by a human operator. In order for the RTUs and the central host computer to exchange data and control commands, there needs to be a communications system to allow messages to be sent/received over large distances, as well as an agreed-on message format and predefined set of messages and responses.

SCADA system designers in the 1960s had to use the currently available long-distance telecommunications technologies of the time, and that meant either the telephone company ("Ma Bell") technology or licensed radio technology. Frequently, the RTU equipment (and the process being monitored and controlled) was located in remote areas where no telephone service was available and possibly too great a distance from the SCADA operations/control center to reliably employ radio communications. In those instances, the process owner (e.g., a pipeline company or an electric utility) would build their own telephone system, using the same equipment that the telephone company would have employed (microwave relay towers, signal multiplexers, etc.).

Analog telephone and analog radio technologies were designed for voice (sound) communications. Thus, SCADA systems required the use of *MODEMs* to turn computer and RTU electrical signals into sounds. In the late 1960s, MODEM technology was restricted to very low data transmission rates, typically 110–300 bits per second. The RTUs in the 1960s were electronic, and constructed with digital components, but not computer-based, so they had to be hardwired to support a simple set of messages for exchanging data with the central host computer.

Further, since most communications would take place at 1200 bits per second (bps) or less, keeping messages short was essential. The set of messages that could

be sent to the RTUs and the set of messages that the RTUs could generate together define a *communications protocol*. In the 1960s, vendors of SCADA systems had to design and build their own RTUs and thus also defined their own (proprietary) communications protocol(s). In certain industries today (primarily the electric utilities), there are still RTUs utilizing some of these old, obsolete protocols—often referred to as *legacy* protocols.

In the 1960s, the *universal asynchronous receiver transmitter* (UART) chip had not yet been invented, and microprocessors were just beginning to be invented, so it was up to each vendor to decide the format of their protocol messages (i.e., how many bits in a message). To simplify the electronic design of the RTUs, most vendors elected to send all available numeric (analog) or binary (status) data in single, long, many-bit messages (fig. 1–2). This would mean messages of 74, 96, 123, or some other extended number of bits. In essence, a response to a polling message from the SCADA host was the transmission of the current values of all of the inputs (analog and status/digital), sent as one long message. These early protocols have very few message types and variations (all of which were built into the RTU hardware), and data values were either single bit or an 8 bit binary integer.

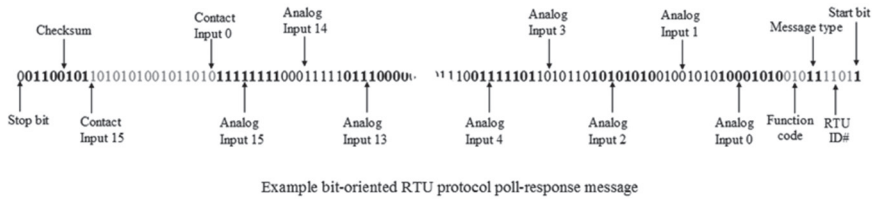


Fig. 1–2. Example bit-oriented message format (starting and ending portions only, owing to actual large number of bits required in a full-length message)

These *bit-oriented* protocols fell out of favor with the invention of the UART chip (and microprocessors), but as previously mentioned, some of these legacy bit-oriented protocols remain in limited use today. These types of protocols usually require the use of specially designed interfaces that can receive and generate the necessary long-bit-sequences. Specially programmed single-board computers are often used for this task. Most 'serial' RTU protocols still used today are based on constructing the messages using some integral number of 8 bit octets/bytes which are suitable for asynchronous serial transmission via UART circuits. Although they normally don't come as a standard interface anymore, computers today can still be equipped with RS-232 serial ports (called 'COM:' ports in the Windows® operating or designated as "ttyS0, ttyS1, etc." in a Linux operating system all of which employ UART circuitry to make them function. Protocols based on messages that use an integral number of octets are generally called *character-oriented* protocols. You will also occasionally hear these two different types of protocols referred to as *synchronous* and *asynchronous* protocols, but this is not technically accurate. In fact,

both are asynchronous (meaning the time between messages is highly variable and so you can't predict when the next one will arrive), but one uses message frames that are an even multiple of octets (8 bits) and the other some vendor-defined frame consisting of a large number of bits (usually several dozens). The differences between these two categories of protocols will be discussed in more detail in a later chapter. Just understand that character-oriented serial protocols can be sent and received with conventional 'COM:' (RS-232) serial ports whereas the bit-oriented (long frame) protocols require specialized hardware for transmission and reception. It was common to use external hardware (a single-board computer) to receive these bit-oriented messages and then break them into octet multiples and send each octet as a separate character into a standard 'COM:' port or a standard "tty#" port. And the process was reversed for transmitted messages.

A SCADA system is used to fetch and present current data values to a human operator. The time required to refresh the measurement data in a SCADA system that represent the *current state* of the remote process, through the polling of RTUs, depends on several factors:

- The bit rate (110, 300, or 1,200 bps) of the polling communication circuit(s)
- The number of RTUs sharing a given communications circuit
- The length (in bits) and number of messages exchanged in the polling process
- The number of communication circuits being used sequentially or concurrently
- The time delay characteristics (latency) of the communication circuits

This is for polling over low-bandwidth serial communication circuits. If high-bandwidth (a.k.a. *broadband*) communications are being used to communicate with field sites, then the most important factor is the last one in the list above.

The protocol used by a SCADA host to communicate with its RTUs can be designed to permit multiple RTUs to share a common communication circuit, much like a party-line telephone circuit (if you're old enough to remember what those were). This means that the protocol incorporates some mechanism (usually an RTU identification number [ID] in the message) that allows a given RTU to identify which messages are intended for that RTU and to ignore messages addressed to other RTUs on a shared circuit. Placing multiple RTUs on communication circuits reduces the required number of such circuits, but it can lengthen the time required to poll all RTUs for their current data values. Most SCADA systems that support multiple polling circuits are designed to poll RTUs on all of these circuits concurrently. Thus, if there are x circuits, the SCADA host can be polling x RTUs concurrently (i.e., one on each circuit).

In some industries, the process dynamics are such that a human operator (or supervisory application program) monitoring and controlling the process through a SCADA system needs to have fresh measurements more frequently than with

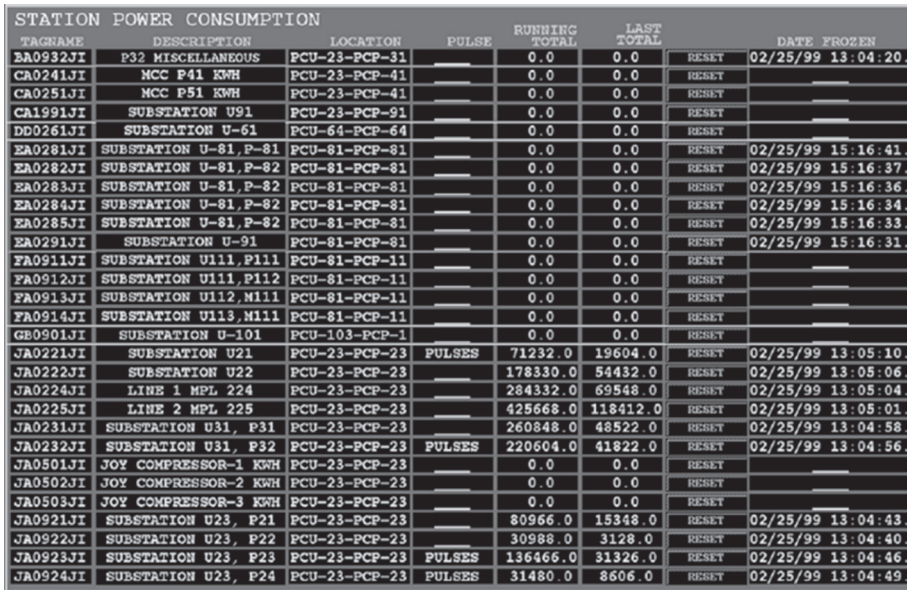
less dynamic processes. In those applications (e.g., electric power transmission), it is common to see multiple communication circuits with little or no *multi-dropping* (sharing) of those communication circuits across multiple RTUs. In less dynamic processes (e.g., water/wastewater transportation), it is not uncommon for all RTUs to be polled on a single, shared radio channel, resulting in much longer times between field data updates.

The presentation of information to a human operator is a fundamental feature of SCADA systems, and in the 1960s, this could be accomplished in several ways. The high-resolution video, projection, and flat-panel display technologies of today were not available to the system designers of the 1960s. In fact, little commercially available display technology existed. Therefore, SCADA vendors were once again required to develop their own HMIs.

A common approach for information presentation was to use a *map board* (also called a *mimic panel*), a wall-sized drawing of the process with indicator lights and numeric meter-style readouts mounted on the wall display at appropriate positions representing physical process areas. Map board technology evolved from the early days of pre-computer SCADA where the MTU actually had outputs (analog and status) that were driven by values received from the RTUs. Essentially, the map board provided an overview of the distributed process. By the 1980s and 1990s, the lights and numeric displays of the map board were electrically controlled by I/O signals from the host computer and updated based on data received from the RTUs. In many SCADA system control rooms today, you will still find the equivalent of a mimic panel/map board display but implemented with modern video projection or high-resolution flat-panel (*video wall*) display technology.

Information presentation in the 1960s would also have included printed reports, logs, and alarm messages. It might also have included very basic *cathode-ray tube* (CRT) displays with tabular, alpha-numeric information (fig. 1–3) or very primitive and simplistic, monochromatic, semi-graphic displays (similar in nature to fig 1–4). Such CRT-based displays would probably have used hardware and software developed by the SCADA system vendor and would have been monochromatic (black and white). As a general note, these early SCADA systems were programmed in assembly language (with an occasional application written in Fortran), and essentially 100 percent of the software (and a good bit of the hardware) was proprietary too, and developed and manufactured by the SCADA vendor, including the operating system software for the hosts and the RTUs.

It should be noted that these examples of early CRT displays are actually more advanced than was generally available in the 1960s. Video display technology was in its infancy and the technology available for commercial applications (as opposed to military applications) was low resolution, monochromatic, character-oriented and fairly expensive. Figures 1–3 and 1–4 are intended to convey the general idea of the style of such displays, but these images actually come from systems of an early 1980s level of technology. Actual images from the 1960s would only survive as photographs, since no digital image capture technology had yet been invented.



STATION POWER CONSUMPTION

TAGNAME	DESCRIPTION	LOCATION	PULSE	RUNNING TOTAL	LAST TOTAL		DATE FROZEN
BA0932JI	P32 MISCELLANEOUS	PCU-23-PCP-31		0.0	0.0	RESET	02/25/99 13:04:20.
CA0241JI	MCC P41 KWH	PCU-23-PCP-41		0.0	0.0	RESET	
CA0251JI	MCC P51 KWH	PCU-23-PCP-41		0.0	0.0	RESET	
CA1991JI	SUBSTATION U91	PCU-23-PCP-91		0.0	0.0	RESET	
DD0261JI	SUBSTATION U-61	PCU-64-PCP-64		0.0	0.0	RESET	
EA0281JI	SUBSTATION U-81,P-81	PCU-81-PCP-81		0.0	0.0	RESET	02/25/99 15:16:41.
EA0282JI	SUBSTATION U-81,P-82	PCU-81-PCP-81		0.0	0.0	RESET	02/25/99 15:16:37.
EA0283JI	SUBSTATION U-81,P-82	PCU-81-PCP-81		0.0	0.0	RESET	02/25/99 15:16:36.
EA0284JI	SUBSTATION U-81,P-82	PCU-81-PCP-81		0.0	0.0	RESET	02/25/99 15:16:34.
EA0285JI	SUBSTATION U-81,P-82	PCU-81-PCP-81		0.0	0.0	RESET	02/25/99 15:16:33.
EA0291JI	SUBSTATION U-91	PCU-81-PCP-81		0.0	0.0	RESET	02/25/99 15:16:31.
FA0911JI	SUBSTATION U111,P111	PCU-81-PCP-11		0.0	0.0	RESET	
FA0912JI	SUBSTATION U111,P112	PCU-81-PCP-11		0.0	0.0	RESET	
FA0913JI	SUBSTATION U112,M111	PCU-81-PCP-11		0.0	0.0	RESET	
FA0914JI	SUBSTATION U113,M111	PCU-81-PCP-11		0.0	0.0	RESET	
GB0901JI	SUBSTATION U-101	PCU-103-PCP-1		0.0	0.0	RESET	
JA0221JI	SUBSTATION U21	PCU-23-PCP-23	PULSES	71232.0	19604.0	RESET	02/25/99 13:05:10.
JA0222JI	SUBSTATION U22	PCU-23-PCP-23		178330.0	54432.0	RESET	02/25/99 13:05:06.
JA0224JI	LINE 1 MPL 224	PCU-23-PCP-23		284332.0	69548.0	RESET	02/25/99 13:05:04.
JA0225JI	LINE 2 MPL 225	PCU-23-PCP-23		425668.0	118412.0	RESET	02/25/99 13:05:01.
JA0231JI	SUBSTATION U31, P31	PCU-23-PCP-23		260848.0	48522.0	RESET	02/25/99 13:04:58.
JA0232JI	SUBSTATION U31, P32	PCU-23-PCP-23	PULSES	220604.0	41822.0	RESET	02/25/99 13:04:56.
JA0501JI	JOY COMPRESSOR-1 KWH	PCU-23-PCP-23		0.0	0.0	RESET	
JA0502JI	JOY COMPRESSOR-2 KWH	PCU-23-PCP-23		0.0	0.0	RESET	
JA0503JI	JOY COMPRESSOR-3 KWH	PCU-23-PCP-23		0.0	0.0	RESET	
JA0921JI	SUBSTATION U23, P21	PCU-23-PCP-23		80966.0	15348.0	RESET	02/25/99 13:04:43.
JA0922JI	SUBSTATION U23, P22	PCU-23-PCP-23		30988.0	3128.0	RESET	02/25/99 13:04:40.
JA0923JI	SUBSTATION U23, P23	PCU-23-PCP-23	PULSES	136466.0	31326.0	RESET	02/25/99 13:04:46.
JA0924JI	SUBSTATION U23, P24	PCU-23-PCP-23	PULSES	31480.0	8606.0	RESET	02/25/99 13:04:49.

Fig. 1–3. Example tabular operator display

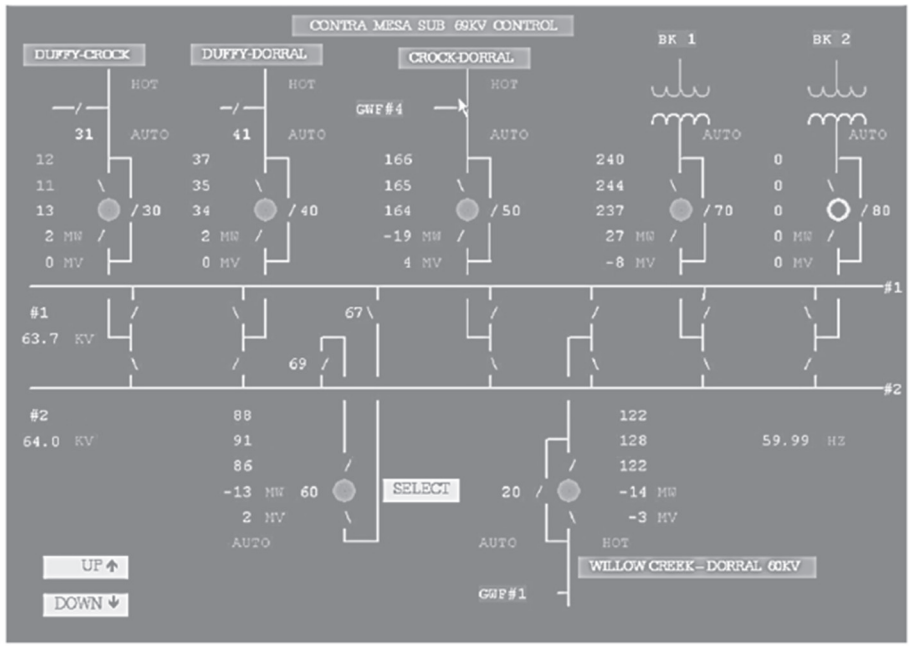


Fig. 1–4. Example semi-graphic operator display

Minicomputers and Microprocessors

The introduction of mainframe computers into telemetry systems, to create the first SCADA systems, was a significant technological advance, because custom application programs could then be developed to make use of the real-time process data—for trending, performing reasonability/alarm checking, and generating additional information by making calculations, as well as for presenting the data to the operator in a variety of formats. The biggest drawback was that mainframe computers were very expensive and not very reliable (so you needed a redundant set, which increased the cost even more). Thus, SCADA systems were restricted to applications that could justify the high implementation costs involved.

The next two major advances in SCADA systems technology were the introduction first of the (relatively) low-cost, 16 bit "*minicomputer*" in the 1960s and then sixteen bit microprocessors, in the 1970s. The minicomputer was a much more cost-effective computing platform as compared to the mainframe computers of the time. They were also much simpler devices and were easier to cobble together into a redundant configuration. By using minicomputers as the host computers, SCADA system vendors were able to drastically reduce the cost of a system. Thus, many more applications of SCADA technology could be financially justified.

It was also common to see hybrid systems—in which redundant minicomputers performed the real-time polling of the RTUs and provided the operator display and remote control functions and a nonredundant mainframe computer was used to run nonessential advanced applications and models. (Of course, the communication hardware and software to connect the two together was also custom designed and built by the SCADA vendor.) As with mainframe computers, the software that ran the minicomputer-based SCADA systems was generally written in assembly language and was usually 100 percent proprietary to the system vendor. Eventually minicomputers evolved into "super" (32 bit) minicomputers, and computer vendors started providing general-purpose operating systems that were suitable for real-time applications and operating system interoperability standards (e.g., Portable Operating System Interface [POSIX] and X-Windows) were developed.

In parallel with this activity, the personal computer ('PC') came into being and ultimately evolved into the powerful devices we have today, with commercially available operating systems, networks, relational databases, peripherals, and much more.

The 8 bit microprocessor revolutionized the design of RTUs in the 1980s, because RTUs based on microprocessor technology could be (re)programmed to add to or expand their functions, perform much more sophisticated functions, and even provide local, closed-loop control.

Occasionally pre-microprocessor RTUs have been referred to as *dumb* remotes, and those with microprocessors have been called *smart* remotes. That 'smart' delineation has become blurred, as the computing power of RTUs has kept pace with

overall microprocessor technology. A modern RTU, with a 32 or 64 bit CPU, could be considered brilliant in comparison to the earliest 'smart' 8 bit microprocessor models.

One major difference between smart and dumb RTUs was in the sophistication of communications protocols they could support. To perform remote control functions with a dumb RTU, SCADA systems often employed what is called a *select-check-operate* message sequence. Messages sent between the host computer and the RTUs were subject to interference and distortion from any number of environmental or communications system sources. Protocol messages are, in actual form, just a string of binary numbers represented by 1s and 0s. If a message were electrically distorted (i.e., if one or more bits were changed in transmission), that could mean that a command that was supposed to operate device X would appear to the RTU as a command to operate device Y.

The error-checking and detection capabilities of pre-microprocessor RTUs was very limited and not always reliable. To prevent a communications error from causing an improper or dangerous control action, early RTU protocols often required a strict sequence of multiple messages (called a *check-before-operate* or *select-check-operate* sequence) in order to affect a control output. In this case, the human operator was an integral part of the message error-checking process. In a select-check-operate protocol, several messages and responses are exchanged in a defined sequence:

1. The operator sends a "select control output n" message to the specific RTU.
2. The RTU returns a "control output n is selected" message and starts a countdown by a *dead-man timer* in the RTU.
3. The operator sends a "prepare to perform operation x on selected output" message to the RTU.
4. The RTU returns an "operation x is to be performed" message.
5. The operator sends a "perform the operation" message to the RTU.
6. The RTU performs operation x on control output n and returns an "operation successful" message to the operator (and also cancels the dead-man timer).

At any point in the sequence, if the operator doesn't like what the RTU says (e.g., if it responds that it has selected output #10, rather than #8 as requested, or if a different RTU responds), the operator can send a "cancel" message—or just let the dead-man timer expire, which will cause the RTU to automatically cancel the activity. The human operator provided message verification and error checking through this process.

With the introduction of microprocessor-based RTUs, it became possible to implement protocols that incorporated comprehensive message error-checking and -correction schemes, thus eliminating the need for operator intervention to validate messages. (However, in the electric utility industry, it is still common to

require that the operator verify control actions in this manner.) Having reliable protocols is even more important when supervisory control commands are to be issued automatically by application programs running in the host computer; operator interaction would not be practical in those instances. With microprocessor-based RTUs, a whole range of additional capabilities were introduced, but a very significant one was having the ability to *download* parameters, program logic, and calculations into the RTU, via the communications (polling) channel. This capability enabled remote modifications of the RTU's functionality, although it required implementing a more sophisticated RTU protocol that included message types for performing such downloading or parameter-modification functions. Over time, as SCADA vendors disappeared, manufacturers of RTU equipment would begin offering to implement the various SCADA vendor's legacy protocols in their RTU products, but usually not to the extent of supporting remote configuration capabilities. Later that was also true for manufacturers of PLC equipment.

Central Architectures

Starting in the 1960s and continuing into the early 1980s, the vast majority of SCADA systems used a central architecture in which a single central computer was responsible for managing and performing all of the functions, including RTU polling, data processing, display generation, report generation, data archiving, and running application programs. Because of the questionable reliability of computers, these centralized architecture systems invariably employed a second, identical (*redundant*) computer, and the two computers maintained some form of on-going dialog to keep the redundant computer synchronized and updated (fig. 1–5).

In a redundant computer configuration, one computer would be designated as the *primary* unit and the other as the *backup* unit. The primary unit would perform all of the SCADA functions and then transfer updated information to the backup computer, so that on any failure of the primary unit, the backup unit could take over and continue operations. The goal of a redundant design is to render a computer failure as transparent as possible to system users (with no loss of data, no loss of control capacity, no missed alarms, no loss of application programs, etc.).

Making redundancy work successfully was the biggest challenge to most SCADA system vendors. Early schemes that involved high-speed copying of memory areas containing current data from the primary to the backup computer (via *direct memory access* [DMA]) would instantly corrupt the backup if the memory of the primary became corrupted. Redundancy schemes were often complicated because of the need to transfer communication circuits, key peripherals, and operator console hardware from the primary to the backup computer. These early systems often used electromechanical transfer switches to implement such peripheral transfers. In other instances, the design of shared equipment, such as operator

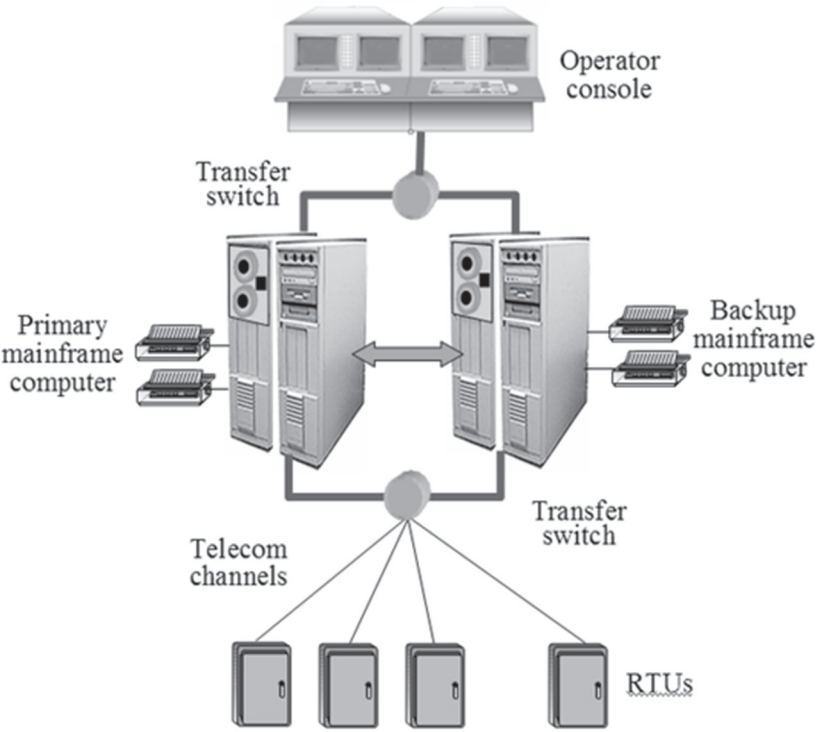


Fig. 1–5. Centralized, redundant SCADA system architecture

video displays, would incorporate a *dual-ported* capability, so that either computer could interact with those devices.

For the triggering of an automatic transfer to the backup computer, some means was needed to automatically determine that the primary was inoperative. Most systems included a special hardware device/circuit called a *watchdog timer*—a circuit that would generate a trigger signal, when a countdown by its hardware timer reached zero, unless that timer was constantly reset by commands from the computer. Both the primary and backup computers would incorporate a watchdog circuit, and each would have an application that was run periodically to reset their respective timers. If either computer stalled, crashed or had a hardware failure, or if its programming went into an *infinite loop*, the watchdog would count down to zero and generate an electrical signal (a hardware *interrupt*) that would cause the transfer of peripherals and notify the alternate computer to take over operations. At least, that is what was supposed to happen. In reality, such redundancy schemes often failed to operate or mis-operated (e.g., triggered a transfer when the primary computer had not actually failed).

Many redundancy designs were implemented, but few worked as advertised. One measure of the quality of a redundancy scheme was the time lag between the data in the primary system and that of the backup. Some vendors claimed that the backup would only lag the primary by seconds; others updated the backup only every few minutes. Another differentiation in redundancy designs was their classification as fully automatic, bump-less, or hot-standby schemes, as compared to offering a warm-backup or cold-backup scheme. The last two of these schemes are less than perfect and might possibly even require some level of manual intervention. A cold-backup is essentially just spare hardware that would be booted-up and begin operating without any of the information that had been in the primary system up to the point of its failure. One reason that RTUs often supported a full-report function was to allow a cold-started backup host to get a set of current field values as quickly as possible.

In a centralized SCADA system architecture, a single computer is doing all of the work of polling RTUs, processing and alarming the incoming data, generating and updating operational displays and reports, performing calculations, and running application programs. Through all of this, that single primary computer still had to allocate central processing unit (CPU) resources and time to update the backup computer with the newest data. Invariably, the number of RTUs, I/O points, calculations, and other functions grew beyond the initially implemented design capacity, and the redundancy scheme would be taxed and degraded and the operational performance of the primary would decrease (e.g. displays would take longer to be presented, alarms were delayed, control actions took longer to perform). With a centralized design, the only way to add capacity to the system was to replace the existing computers with newer, more powerful models. This is one of the primary reasons why a popular replacement design eventually supplanted centralized architecture.

Distributed Architectures

Starting in the 1980s and up to the present, the majority of SCADA systems followed the trend of IT computer technology and went to a distributed architecture in which multiple computers were networked together on a high-speed LAN and specific functions were assigned/distributed to particular computers. The introduction of local area networking (LAN) technologies (specifically Ethernet—IEEE 802.3) in the 1980s made this possible and practical.

One of the most common architectural strategies was to separate the RTU polling process, placing this function in dedicated *front-end* computers (fig. 1–6). This strategy also simplified the problem of computer redundancy, because front-end computers could be stripped down to their basics—RAM, serial ports, and LAN controllers—making primary-backup updating much less complicated. As a given front-end computer received data from RTU polling, it could send data-update

messages to all other computers on the LAN, thus keeping them data synchronized. This was similar to what we now refer to as a 'publish-subscribe' data distribution mechanism.

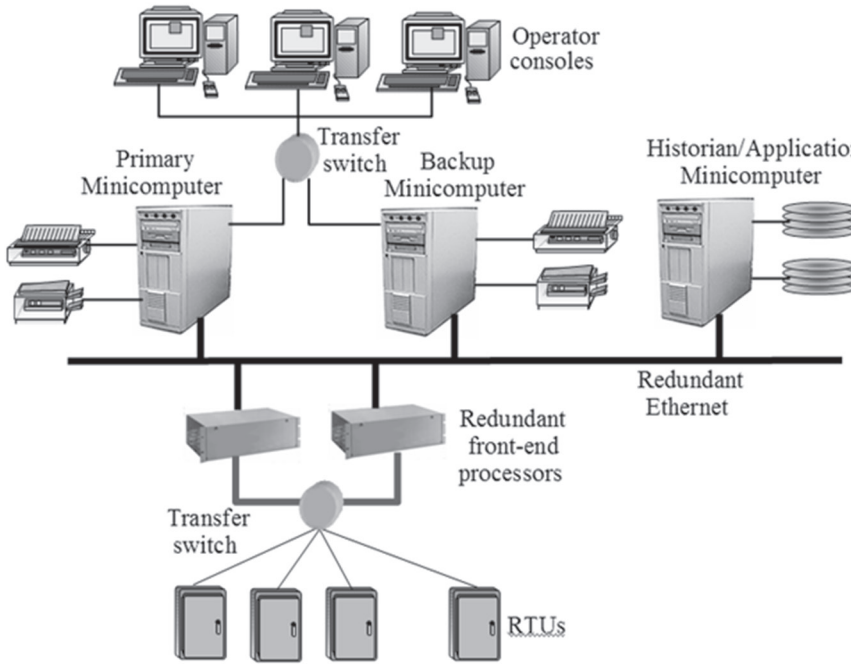


Fig. 1–6. Distributed SCADA system architecture

Another distributed-architecture strategy was to use a separate, dedicated computer to run large, complex, processor-intensive applications. This computer might not be made redundant, because a temporary loss of those applications would not affect the operator's basic ability to control and monitor the process. Historical archiving of key process measurements (trending) became an important factor in SCADA evolution. Rather than equipping every operator workstation with large amounts of bulk storage, it became typical to assign such archiving to a separate computer (a.k.a. a *historian*), and just provide that one historian with large bulk storage capacity (and some means for dumping data to removable storage for offline archiving purposes.) Because of LAN data speeds it was possible for that historian to deliver trend data to operator displays, and other applications, in near-real time, when required.

A distributed architecture was the next logical step from a centralized design, because it increased overall computing power by spreading the various SCADA functions into additional computers dedicated to performing those specific

functions. It also improved overall system reliability by making it possible to lose only certain SCADA functions, as a result of a fault or malfunction in one of the computers, rather than losing the total system. This ability to suffer a partial functional loss but to maintain critical operations is usually described as *graceful degradation*.

Client/Server Designs

In the 1990s, with the advent of powerful 32 bit microprocessors and personal computers—as well as Transmission Control Protocol/Internet Protocol (TCP/IP) networking technology, high-speed Ethernet, and commercial real-time operating systems—the design of SCADA systems took another evolutionary step. In effect, SCADA systems began to look very much like the corporate information technology (IT) business systems of the time. These new SCADA systems were designed to be *scalable* and *fault tolerant,* and the software that performed the SCADA-specific functions was redesigned and rewritten for distribution across multiple computer *servers*. Software that had been monolithic in prior generations (e.g., operator console display software) was broken into interacting components that could be run in the same computer or in different ones, with cross-LAN communications. The need for peripheral transfer switching and complex redundancy schemes was replaced with smarter software. *Clients* (e.g., any program that generates operational display updates and requires data sources to do so) could seek out their respective servers (e.g., an RTU polling program) across the LAN to get the required data. If for some reason the server became unavailable, these clients could automatically switch to an alternative server if one was available. If more computing capability were required, additional servers could be connected to the high-speed network, and more copies of the client/server software could be loaded and run (fig. 1–7). Distributed client-server architectures still remain the basic SCADA system design architecture today.

By the late 1990s SCADA was becoming a special set of applications that could be run on any conventional, commercial computing platform. In fact, there were several commercially available SCADA software packages (e.g., the FIX® and Wonderware®) that can be purchased as packaged software and loaded onto conventional personal computers (PCs) and servers, as a do-it-yourself SCADA system. One of the major developments that came out of the migration toward the client/server design was a set of standards that define a mechanism for exchanging data between clients and servers developed by different vendors. The Object Linking and Embedding (OLE) for Process Control (OPC) standards were co-developed by a group of vendors (including Microsoft) and end users and currently form the basis for a large number of commercially available SCADA software products. Unfortunately, the original OPC specifications were based on integral functionality in the Windows operating system, specifically DCOM (distributed component object model) and RPC (remote procedure calls) both of which were

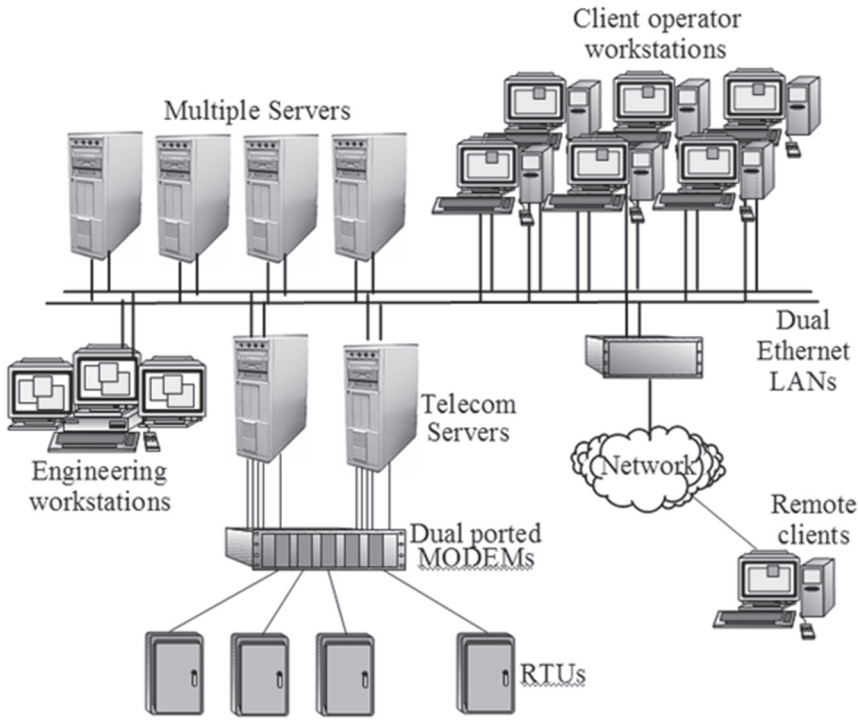


Fig. 1–7. Client/server SCADA system architecture

eventually discontinued by Microsoft in the early 2000s. As a result, a revised standard, called OPC-UA (OPC Unified Architecture) was developed and introduced in 2006 as a replacement and is currently being built into a wide range of automation products. The original OPC is now referred to as "OPC Classic" and there is a large installed base of products and systems that use that technology. OPC-UA has gained in popularity because it is platform agnostic, it does not require a Windows operating system to function.

Technological Convergence

Because SCADA systems are based on computer technology, their designs have evolved in step with advances in computer technology. This also means that as computer technology has been expanded and evolved —to include, for example, Ethernet LANs and wide area IP-based networks (WANs), PC multi-media workstations, high-resolution video displays, standard commercial operating systems and massive memory and bulk storage capacity—the designers of SCADA systems have incorporated these technological enhancements into their system architectures. In comparison with the SCADA systems of the 1960s, those produced

today have undergone a huge decrease in the percentage of proprietary hardware (excluding RTUs) and software (including operating systems, HMIs, and network software) used in creating these systems (see table 1–1).

Table 1–1. Trends in the evolution of system architecture

	1960s	1970s	1980s	1990s	2000s
Proprietary hardware	75%	50%	30%	10%	5%
Proprietary software	100%	90%	75%	40%	25%

One of the biggest changes in SCADA is that, unlike vendors in the 1980s who were system integrators that provided turn-key solutions and even field installation support, most of the SCADA vendors today merely sell software licenses and product training and technical support. It is common today for the corporate IT department of an operating company to purchase and stage all of the basic computer equipment, networking elements, and commercial software required to run the SCADA software. If the corporation has chosen to extend the corporate IP-based WAN out to field sites, then the corporate IT department may also be responsible for the field equipment. This approach can, in fact, lead to treating SCADA functions as just another business application and running the SCADA software on a virtual machine provisioned within the corporate IT server farm, as shown in Figure 1–8.

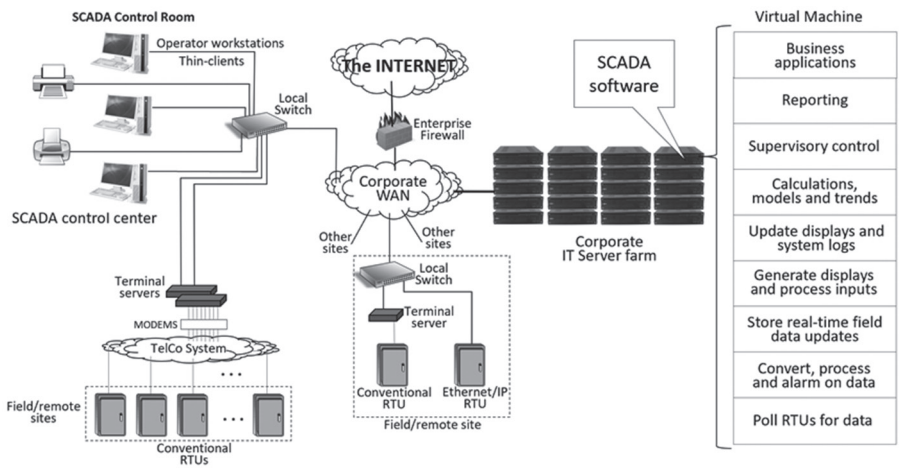


Fig. 1–8. Virtualized SCADA system implementations

There have been significant commercial benefits to this trend: SCADA systems cost much less to construct today (regardless of how much a vendor may charge you to purchase their software); they are interoperable across vendors and with other types of computer systems; it is easier to find technical personnel with relevant programming and operating system skills; and there are many more off-the-shelf

applications available from third-party suppliers. That is the good news. The bad news is that all of this has made the SCADA systems of today far more susceptible to malware, hackers, and cyberattack.

Today, most SCADA systems are constructed using multiprocessor/core servers based on the Intel or Intel-compatible x86 family of microprocessors. Modern SCADA systems use a distributed client/server design, with all computers and PCs connected via a high-bandwidth Ethernet LAN (either redundant or made fault-tolerant via Rapid Spanning Tree) and TCP/IP networking—and will be running either a Microsoft or a Linux variation operating system in each computer server and PC. Today, they may incorporate virtualization technologies as well. They may use a standard, commercial relational database package (e.g., SQLserver®, Oracle®, or the OSI-PI® package) as their data historian. They probably provide the human-machine (operator) interface on standard PCs using SCADA software that either runs as a Web server or uses the graphical displays and presentation capabilities of the SCADA package itself. A modern SCADA system may support the OPC "classic" standard (or even OPC-UA) and use it for inter-system data exchanges. The point is that most of this is the same technology taught in every college and university computer science and IT curriculum and that is being used for office automation, on every desktop, and in most homes.

Moreover, this is the technology that is well known to the hackers of the world. A SCADA system from the 1960s and even the 1970s or early 1980s would probably be relatively safe from a cyberattack, because of the proprietary operating systems, communications protocols, and operator display technology. However, they would also be incapable of providing the interoperability, connectivity, and compatibility of current systems. We can't go back to obsolete technologies; thus, we have to make design changes to modern SCADA systems, so that they are secure from hackers and cyberterrorists.

Ubiquitous Internet and IP Networking

In the last three decades, as the Internet and IP networking have spread to all points of the compass, the widespread availability of these high-speed, wide-area networking capabilities has enabled some unusual experiments in SCADA technology. Many small, rural utilities that provide electrical, gas, and water distribution do not have the manpower, expertise, or resources to purchase and stage and support their own SCADA systems, and they may only have a handful of field locations that they would like to be able to monitor and control. So, purchase and maintenance of their own SCADA system makes no financial sense, even if it would be operationally desirable. There have been some experiments with sharing a SCADA system wherein multiple such utilities, regardless of proximity or location, can pay a small up-front, and thereafter monthly, fee and have the benefits of a SCADA system without all of the headaches. Figure 1–9 shows a simplified diagram of the concept. A SCADA

service provider sets up a system and provides each participating utility a Web-based portal that provides them with displays of their own information (but isolates them from the information of others.) The SCADA vendor provides the equipment to the utilities and assists them with the initial configuration and installation of RTUs and workstations and setting up of their basic displays, alarms, and control screens. Each field site needs to have power and network connectivity, which can be arranged through local telecommunication providers. In some instances, the communications may be via cellular connectivity or possibly even via satellite link. The point is that by sharing the cost of running and maintaining a SCADA system among a fair number of utilities, this can be profitable for the SCADA service provider and cost-effective for the various utilities. This is a form of SaaS (SCADA as a Service).

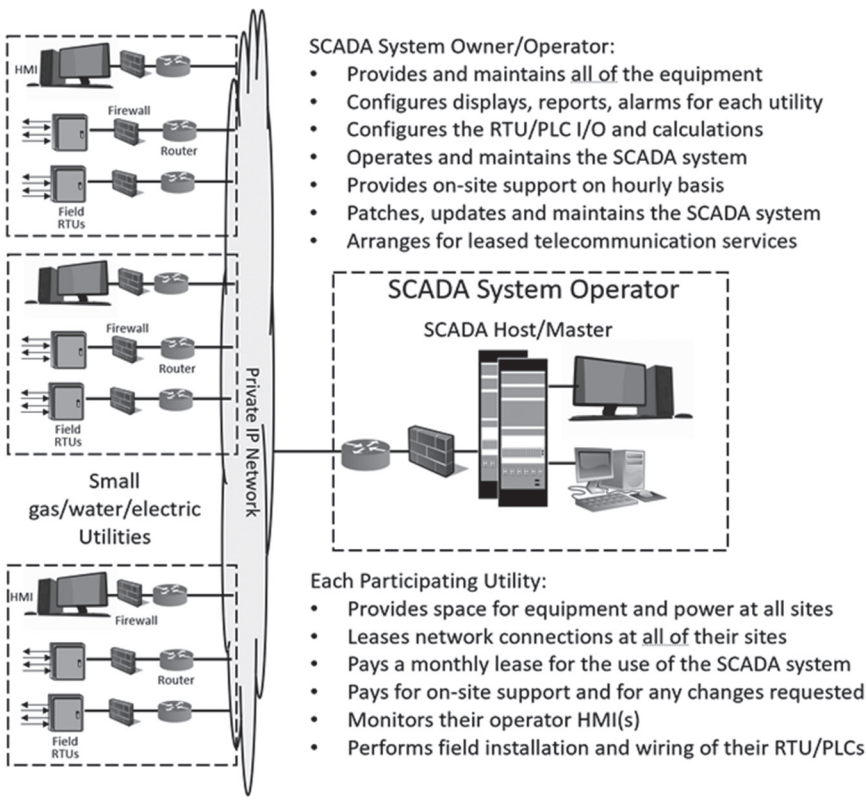


Fig. 1–9. SCADA as a service

These days, there is a lot of noise made about the evolving and emerging Industrial Internet of Things (IIoT) and how it will impact automation technologies (of which SCADA is a principle one). A lot of the IIoT discussion revolves around embedding intelligence into the lowest levels of automation, which for SCADA

systems often means the RTUs and field equipment. Another major discussion point is the ability to, in theory, have real-time Internet connectivity anywhere and everywhere. As has been mentioned, modern RTU products (including PLCs) now have a massive amount of computing power and memory due to advances in microprocessors and memory technology. It is quite reasonable to envision RTUs that are fully autonomous and capable of offering a sophisticated Web-based interface to anyone authorized for such access. (RTUs and PLCs capable of generating Web displays have been around since early 2000.) Web 2.0 technologies such as JASON are well-known, in common use for commercial Web-based products, and when properly used, can offer a rich, interactive user experience. It is quite reasonable to foresee a time where SCADA functionality requires no actual SCADA system, just broadband Internet access and a good Web browser. Figure 1–10 shows how such an arrangement might be configured. In effect, all of the operational display, measurement processing and alarming, historical trending, and even report generation could be pushed down to the RTU level. The RTUs could even directly offer data access to other systems via OPC-UA connectivity. Clearly, there would be installation and configuration requirements for the "brilliant" RTUs, but that is necessary for even just smart RTUs. Clearly in such an arrangement there would be a need for top-level cybersecurity measures to protect the RTUs and control cyber access.

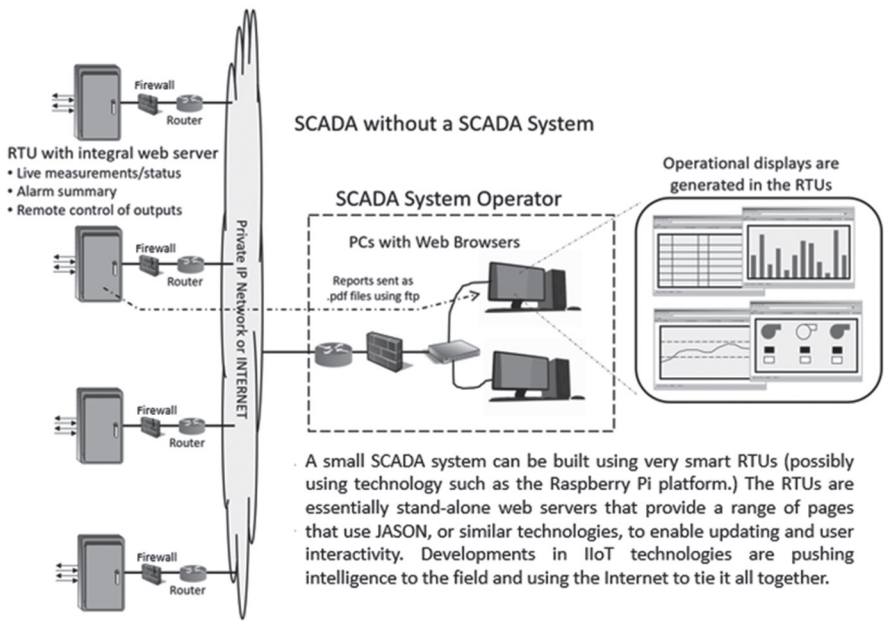


Fig. 1–10. SCADA without a SCADA system

Generalized Software Architecture

The technology underlying SCADA systems has evolved in lockstep with corresponding advances in digital electronics, communications, and computing. The architectural designs of SCADA systems have been changed to take advantage of these new technologies. Regardless of the hardware architecture and computer, networking, and operating system technology employed, SCADA systems still perform a basic set of functions:

- They communicate with field-based outstations and subsystems for retrieval of up-to-date process and plant information.
- They maintain a constantly updated set of this information in a database.
- They provide an interface that enables human operators to examine and display this information.
- They provide some mechanism for a human operator to affect a process change by sending control commands back to the outstations and subsystems.

But it would be a pretty poor SCADA system, however, that didn't provide at least a few additional functions, such as the following:

- Automatic comparison of field measurements against user-defined alarm limits and against their prior values.
- Automatic annunciation and logging on detection of alarms and events.
- Long-term recording of the value/status history of selected field inputs and calculated values.
- The generation of user-defined reports and logs that incorporate available measurements, calculations, and historical/recorded data.
- The calculation of important data values based on user-defined equations that incorporate available historical and real-time measurements and possibly even user inputs.

All modern SCADA systems support the features and functions listed previously, and most support a whole lot more, including industry-specific applications that customize the SCADA system for the unique requirements of individual market segments. Figure 1–11 provides a simplified information flow diagram for the basic functions and features of a generic SCADA system. The actual implementations (where the particular functions occur, where data is located, how the bits and pieces are connected and interoperate) will depend on the specific architectural and technology choices made by the individual SCADA system vendors.

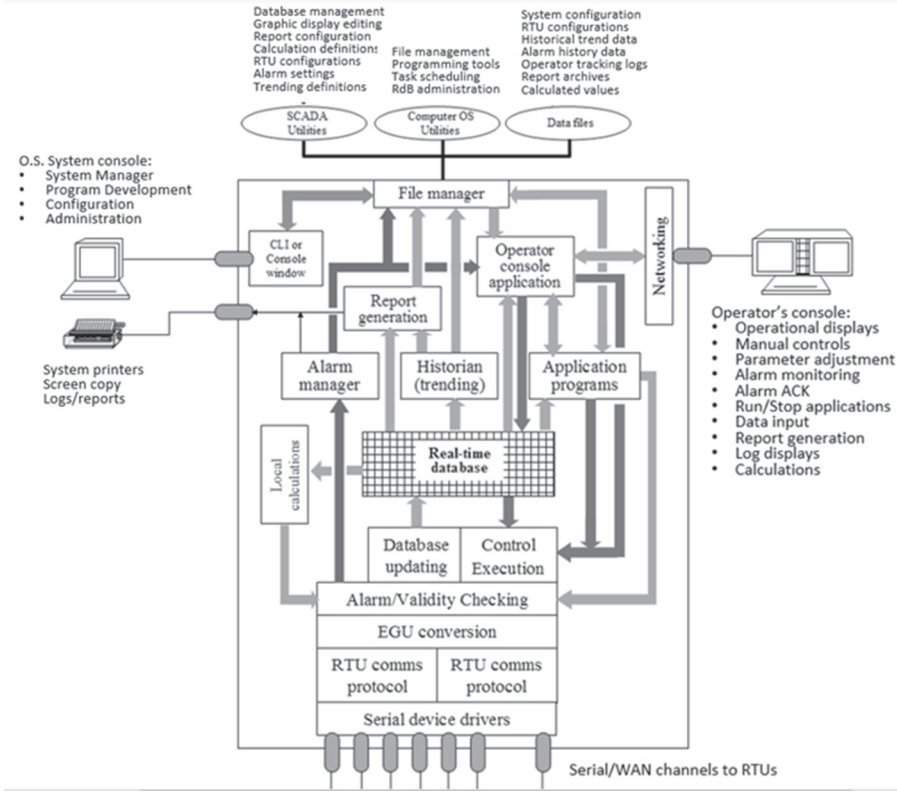


Fig. 1–11. Generalized Information Flow within a Generic SCADA System

2

Remote terminal units

Basic Features and Functions

As previously mentioned, the purpose of a SCADA system is the remote sensing (viewing) and control of geographically dispersed processes. (If it doesn't support remote control, then it is actually just a DAS—a data acquisition system—and not a SCADA system.) To achieve this aim, a typical SCADA system incorporates multiple RTUs, physically placed in the field at key measurement and control locations. The RTU began as a hardwired, fixed-function electronic device that could respond to simple commands received as serially transmitted numeric (binary) codes. Early RTUs were used for remote telemetry purposes, even before SCADA systems were created through the integration of a central computer.

RTUs were connected, through low-speed serial communication channels (e.g., leased analog telephone circuits), to *master terminal units* (MTUs), electronic devices that provided an interface for a human being, so that data from the RTUs could be displayed and control outputs of the RTUs could be manipulated. An MTU generally consisted of an alphanumeric readout or display and a keypad for entering parameters and commands (fig. 2–1). An operator could enter an RTU ID and an input number and press a poll key. The MTU would send a serial numeric code (message) to the RTU(s), receive the response from the RTU (in the form of a coded numeric message), and display the value of the requested input point on its readout. RTU control outputs could be manipulated in a similar fashion. More sophisticated MTUs supported multiple numeric readouts and continuous, automatic polling (with displays that would also cycle through the measurements as they were polled and updated values received). Some MTUs could also be equipped with analog and contact outputs that could be wired to lights and meters on a mimic panel (a.k.a., a map board). As field signals were received, these outputs would be driven to the same values to provide an operational display for field measurements and equipment status. Like their corresponding RTUs, the MTUs were electronic devices with hardwired functions and features, rather than programmable, computer-based devices. (They were "digital" in that they contained logic gates and digital components, but no CPU or stored program logic.) When computers were introduced to replace the electronic-MTUs (and programmed to

replicate the communication protocol messages), the term MTU was often thereafter applied to the computer. You will still hear SCADA central computers called the "host," the "central unit," or occasionally even the "MTU" or just the "Master."

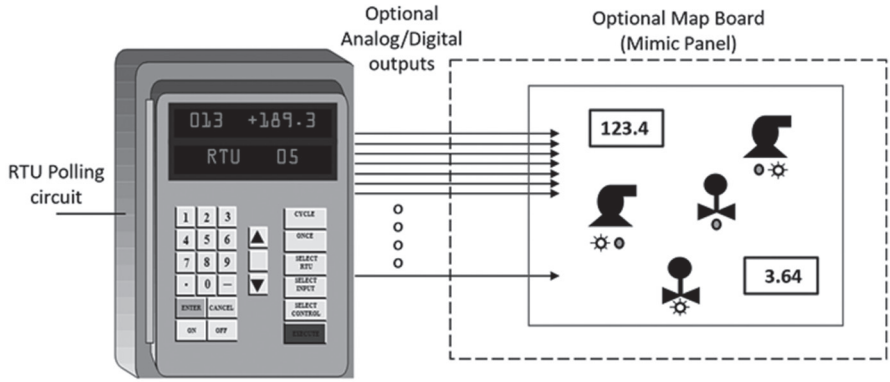


Fig. 2–1. Typical MTU console

When the first SCADA systems were developed, the obvious strategy was to replace the MTU with a computer, programming that computer to send and receive the same simple numeric messages that had previously been generated and processed by the MTU. The set of messages to and responses from the RTUs constitutes a communications protocol. Because the early RTUs and MTUs were electronic, and even digital, but not based on any form of programmable computer technology, they had to be designed with electronic circuits that could generate and interpret these simple numeric/binary message codes. As might be expected, this kept the number and complexity of such messages to a minimum. The only functions supported by early RTUs were reading and transmitting the current value of analog and status inputs and generating analog and contact outputs in response to a properly formatted control message. Somewhat surprisingly, given the pace at which other technologies have changed since the 1960s, the types of input and output signals supported by RTUs, until very recently, were limited to the same types of analog, status, and pulse signals as would have been found in pre-SCADA MTU-based systems.

Analog inputs

An analog input signal is generally a voltage or current that varies, over a defined value range, in direct proportion to a physical process measurement. It is very common, in most industrial processes since the 1970s, to use a 4 to 20 milliamp signal to represent physical measurements such as pressure, flow, and temperature. Other industries, such as the electric utilities, use signal ranges like –1 to +1 milliamp or 0 to 1 volt or –10 to +10 volts. (There are also devices that produce

a variable air pressure or fluid pressure signal—which are converted to current or voltage signals suitable as RTU inputs.) In addition, there are special sensory devices, like *thermocouples* and *RTDs* (Resistive Temperature Device), that work in nonlinear, millivolt (1/1000th of a volt)-level signals, which vary with the temperature of the device and require special signal handling and processing.

Inside an RTU, a special circuit called an analog-to-digital (A2D) converter produces a binary number that corresponds to the voltage level applied to the input. Early RTUs generally had 8 bit A2D circuits, which means that they could represent an input voltage range as a binary value from 000_{10} to 127_{10} (0x00 to 0xff hexadecimal). In current RTU technology, 16 bit A2D circuits have become commonplace. The number of bits defines the resolution accuracy available for the reading of inputs. More bits mean more resolution and, in theory, more accuracy—although in some cases that resolution is wasted because the field device producing the measured signal is not very precise.

Because A2D circuits are expensive, the tendency among RTU (and PLC) manufacturers was to design RTUs with only one; this circuit was then shared by *multiplexing* analog inputs, one at a time, with additional circuitry. Multiplexing took time to operate (because early on it was done using electromechanical *reed relays*); thus, if there were a large number of analog inputs, the scan time for reading all of them for their latest value could extend to several seconds or even tens of seconds depending on the input count of the RTU. If the process being monitored by the RTU was highly dynamic, this might not be acceptable; therefore, some RTU manufacturers designed RTUs that could be expanded with multiple A2D circuits. With the advent of microprocessor-based (smart) RTUs, it became possible to configure the RTU to scan selected analog inputs more frequently than others (possibly eliminating the need for additional A2D circuits) and to expand protocols to provide means for fetching critical analog inputs more frequently than noncritical ones (e.g., scan critical inputs every second and others, less critical, only every 10 seconds).

The successful reading of analog inputs requires that the input be free from noise or electrical distortions. Current-loop (4 to 20 milliamp) signals are generally immune to typical electrical noise sources (both common and normal mode), which is one reason for the popularity of this signal standard. Another is that an open circuit could easily be detected, as it would read 0 milliamps. Voltage signals, especially very-low-level voltage signals such as generated by thermocouples, are far more susceptible to noise and distortion (in particular, the ubiquitous 60- or 50-Hertz alternating current [AC] electrical noise); thus, the analog input circuitry design of an RTU had to deal with this noise. Many RTU manufacturers elected not to support RTDs and thermocouples as inputs, instead requiring that the user convert these into a conventional milliamp signal external to the RTU. Other vendors implemented hardware filtering (or with smart RTUs, software filtering), to reduce or eliminate signal distortion. A software scheme commonly used to this date is to sample an analog input value twice and average the two samples. These

two samples had to be taken exactly 8.333 milliseconds apart (or 10 milliseconds for 50 Hz noise), thus placing the two samples 180° apart on the AC sine wave (i.e., so that the AC noise cancelled itself out). RTUs today still use this scheme for noise reduction.

Analog outputs

An analog output signal is generally used to manipulate and adjust the operating point of such process equipment as modulated control valves and variable-speed motors and drives. The process and pipeline industries make extensive use of this type of output, whereas the electric utility industry almost never uses them. Again, the 4–20 milliamp signal type is generally used for analog outputs in most process industries.

Status inputs

A status or contact input is a two-state input generally received as either an actual contact/switch closure or a voltage/no-voltage signal. These inputs provide a simple means for indicating the present status of a two-state device control element: open/closed, on/off, running/stopped. All industrial segments make extensive use of status inputs. In many RTUs (and in all of the dumb ones), status inputs are scanned whenever their current values are requested by the host computer.

In some instances, status inputs are used to *time-stamp* critical events or to differentiate the timing between events. In those cases, it may be necessary for the RTU to continuously scan all of the status inputs, looking for and recording changes. This capability was not possible before the development of microprocessor-based (smart) RTUs. In the electric utility market, it is now commonly required that an RTU scan all status inputs every millisecond and that the RTU keep a record of all changes (time-stamped to a one-millisecond accuracy) between polls by the host computer. This means that the RTU must have an accurate time source and memory in which to record input change events. This capability is generally called *sequence of events* (SOE) recording.

Another issue with status inputs is that the process equipment that produces these contact inputs can be mechanically bulky and subject to vibration and sensor-alignment problems. All of this can mean that a contact signal might not be as clean as the flip of a switch (which really isn't all that clean if you capture the signal with an oscilloscope). Often, an RTU actually records that a status input has flip-flopped between states a few times when transitioning from one state to the other. This is called *contact bounce*, and this condition needs to be handled by the RTU. In dumb RTUs, a simple resistor-capacitor (RC) filter circuit was placed across the contact input, to filter out the bounce. An RC filter circuit 'smoothes' out impulse noise and causes voltage jumps to turn into exponential rises/decays. Only a persistent voltage signal will make it through the filter—and only after it

persists for a given amount of time [the time constant of the RC circuit]. In modern RTUs, which might be performing one-millisecond SOE monitoring on such inputs, *debouncing* is done with software logic, since the use of an RC filter would actually cause a time delay before sensing the input change and thus would result in inaccurate time tagging for SOE purposes. In such logic, the first transition time is recorded, even if it is a momentary transition, and is then assigned as the input's SOE time tag if it does turn out to be an actual state transition and not just noise.

Contact outputs

Contact outputs, or controls, are switch closures generated by the RTU, usually in the form of an electromechanical relay driven on and off by RTU circuitry and logic. Every industrial segment makes use of these types of output. Contact outputs can be broken into two subcategories: latched outputs and momentary outputs. A latched output is one that is turned on or off by the RTU and stays in that state (possibly even if power is removed) until a specific command is given to place it into the opposite state. Momentary outputs are outputs that will be turned on and then back off by the RTU as a single-command operation. The duration of the "on" state may be fixed or variable, depending on the capabilities of the RTU. Dumb RTUs normally had a hardware-set "on" duration that could not be adjusted. Modern RTUs often make the "on" time a user-definable parameter per contact output. The process industries make extensive use of latched outputs (and some momentary ones), whereas the electric utility industry uses momentary outputs almost exclusively. Enabling a contact output to remain in its latched state, even if power is lost (if that is required for safety purposes), can be accomplished with various types of mechanical or magnetic latching designs.

Since contact outputs are often used to operate equipment that also has a local manual control panel, it is not uncommon (particularly in the electric utility market) to see pairs of outputs assigned to control a given device. For local manual control, many types of industrial and electrical equipment have one button for "on" (or "start") and a separate button for "off" (or "stop"). The electric transmission industry uses "trip/close" contact pairs to operate circuit breakers. With some RTUs, momentary contact outputs are grouped into control pairs such that one would be wired to the "on" and the other to the "off" circuitry. The electric utility industry generally utilizes momentary contact outputs in this control-pair manner; in fact, RTUs for that market sometimes include hardware or logic that prevents concurrent control of both outputs in a pair. (You would never press the "on" and "off" buttons at the same time, would you?) When dealing with electric utility SCADA personnel, you will often hear this configuration referred to as *trip-close pairs,* because a very common use of RTUs is the remote control of circuit breakers (which you can either "trip" [open] or "close").

In the process industries, a variable-duration momentary contact output pair may be used to control large motor-operated (or hydraulically operated) valves

(e.g., in a water transmission system or a liquids pipeline). With such valves, there is typically one contact that will drive the valve in the open direction and a second contact that will drive the valve in the closed direction. If such a valve is to be operated in an adjustable mode, the RTU controlling it will need to generate open/close contact outputs of a duration based on the travel rate of the valve (percent per second) and the amount of change in position needed. In these cases, there is control logic in the RTU to compute the necessary contact duration and pass it to the control output software.

Pulse inputs

Pulse inputs are a special class of contact input (although they yield a numeric value like analog inputs) and may be handled by the contact input hardware by use of special software. There are two broad categories of pulse inputs: pulses that are continuously counted (called *accumulators* or *totalizers*) and pulses that are counted over a precise, repeating, time interval. In both cases, the individual pulses generally represent a specific quantity of something: a barrel of oil, a cubic foot of gas, a kilowatt of power. If we want to totalize these quantities, then we just keep counting pulses and multiply by a scalar that converts pulse count to material quantity; by contrast, if we want to know delivery rates, then we count pulses over a fixed time interval to come up with gallons per minute, cubic feet per hour, and so forth.

In the process industries, pulse signals are often used for both purposes, and pulse rates may be as fast as 1.5 kilohertz (1,500 pulses per second) or even faster with measurement devices like turbine flow meters. RTUs normally require special pulse-counting hardware for these types of signals. In the electric utility industry, pulses are often used to totalize power delivery, and pulses are generated by electromechanical power meters at a very slow rate (slower than one pulse per second). With dumb RTUs, this still required special hardware; however, with smart RTUs, no special hardware is required, since the pulsing is slow enough to be sensed by the status input hardware. Many smart RTUs allow individual contact inputs to be designated as status inputs or accumulator inputs under software control.

Pulse outputs

Pulse outputs are a special class of momentary contact output and are typically handled by the contact output hardware and special software. Pulse outputs are a modification of the variable-duration momentary contact output; in place of a contact that is held in the "on" position for a specified amount of time, the output is toggled between "on" and "off" a specified number of times (fig. 2–2). As with variable-duration contact outputs, this is usually implemented in smart RTUs as a software function and doesn't require any special contact output hardware. Pulse outputs were mainly used in the process industries and are rarely used anymore.

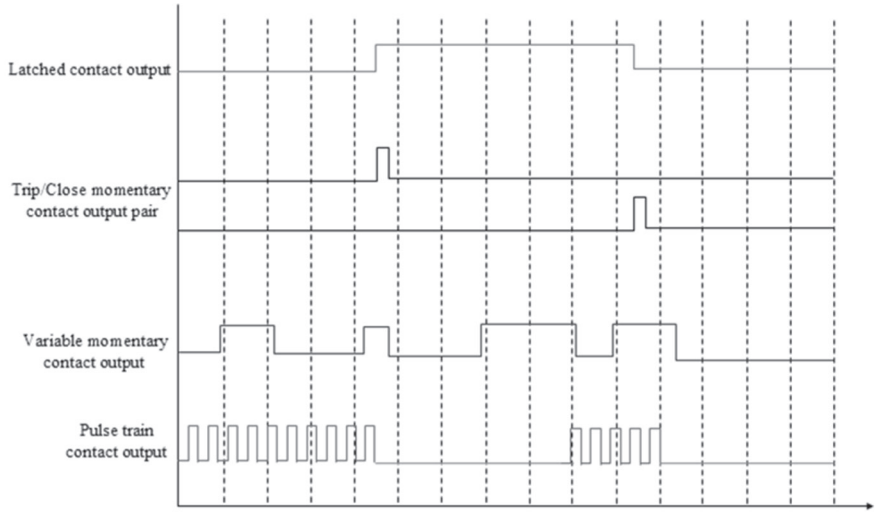


Fig. 2–2. RTU contact output (control) types

Smart RTU Technology

RTUs underwent a fundamental change with the introduction of microprocessor-based units. Prior to that evolutionary step, adding any features and functions to an RTU involved a massive hardware redesign and changes to the protocol messages. No real effort was ever made to do much more with dumb RTUs than just expand the maximum quantity of input/output (I/O) and the types of I/O supported by those RTUs. However, with a microprocessor running the RTU, changes were mainly a function of software modifications, including protocol enhancements. The early smart RTUs used 8 bit microprocessors like the Intel 8080 and 8085. These RTUs were limited in memory capacity (4–16 kilobytes of RAM/ROM was typical) and processing power; still, they were capable of supporting functions that were not possible with dumb RTU technology, such as simple calculations. As microprocessor technology advanced and CPUs became faster and were expanded to include math coprocessors and more memory, it became possible to add more features and functions to the RTUs (fig. 2–3).

RTU feature evolution was not the same across all industry sectors using SCADA technology. SCADA system vendors tended to maintain a primary presence in one key segment (pipelines, electric power, transportation, water, etc.) and focused product developments, including RTU developments, on the needs of that market segment. Many SCADA system vendors that attempted to cross over into other market segments to expand their business base discovered that their systems and RTUs lacked features and capabilities that had become standard in these other market segments.

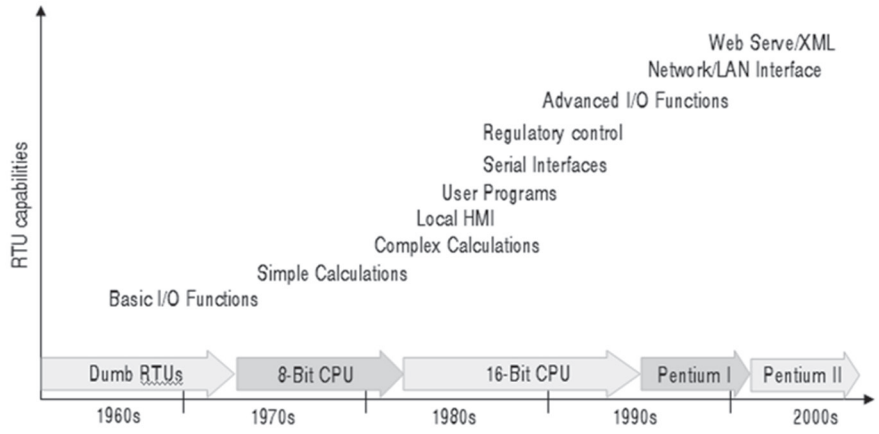


Fig. 2–3. Evolution of smart RTU technology and capabilities

Probably the most advanced RTUs, from a feature and function perspective, are those used in the oil and gas pipeline market. RTUs in the pipeline industry were used to perform local regulatory (e.g., *PID*) and sequential control functions as early as the mid-1980s. These RTUs could download user-defined control schemes and logic over the polling communication channel and could perform advanced calculations, such as the American Gas Association (AGA) volumetric computations (AGA-3, AGA-5, AGA-7 and NX-19). They also had to support serial communications with other smart instrumentation (e.g., a calorimeter or analyzer) to get the data needed to perform some of those calculations. These RTUs, once downloaded with the appropriate logic, would function as independent, autonomous local controllers, able to perform their assigned functions without any operator intervention, even if communications to the host were lost. As memory capacity expanded, some of these RTUs incorporated the ability to buffer data in the event of extended communication outages and then to upload this information to the host when communications were restored.

All of these capabilities required extensive modifications to the RTU communications protocols. Even as vendors moved rapidly to match the capabilities of their competitors, protocol versions were released at an even faster rate—sometimes every few months. If the revisions merely added functions, without changing any preexisting functions, then it was generally possible to mix RTUs with different protocol versions on the same communication channel. However, occasionally a protocol enhancement would modify the protocol sufficiently that such mixing was not viable, and SCADA system owners would have to send technicians to the field, to make *firmware* changes to all of the existing RTUs or at least to those on a shared communication circuit. Keeping track of protocol versions and compatibilities was a major headache for SCADA system owners and vendors.

Over time, it became typical for SCADA vendors to have a wide range of RTUs with differing I/O capacities, features and functions (and prices), and possibly a set of different protocols that would be used, based on the particular RTUs being supplied. Although there have been successful cases of multiple RTU serial protocols coexisting on the same communication channel, this is not recommended (although it is a ploy occasionally used when replacing legacy RTUs). It is all too possible for messages and data sent using one protocol to be accidentally interpreted as a valid message to an RTU on that same polling circuit that uses the other protocol.

Serial ports

RTUs, at least into the early 2000s, normally came with at least one serial port, through which the SCADA host communicates with the RTU. (Today they may still have a serial port or two for communication, but probably also have an Ethernet port and support IP-based protocols.) This serial port is typically dedicated to supporting the protocol-defined polling and command messages received from the host. With the introduction of microprocessor-based RTUs, it also became typical to have a second serial port, dedicated to local interrogation of the RTU for diagnostic and configuration purposes (often referred to as the *console port*).The second serial port was usually managed by a special program in the RTU called a *command line interpreter* (CLI), as in the Microsoft Windows "cmd.exe" command prompt program. The CLI understood simple text commands and provided a service technician with a way to examine and modify internal settings and parameters and to run whatever diagnostic routines were available. A typical use of this console port would be to tell the RTU how to configure the host polling port (in terms of data [bps] rate and number of stop bits, etc.) and its unique ID number; the console port also usually allowed for manual reading and controlling of the RTU's I/O (and today it might be used to configure the Ethernet port and set IP address and subnet mask values).

In some industrial applications, it was occasionally necessary for two separate organizations, with their own SCADA systems, to need the same field measurements. Initially, this meant placing two separate RTUs at the same location and wiring the same inputs to both units. This worked well enough for inputs, but if both organizations needed to control the same field equipment, double wiring presented a problem. The solution eventually developed was to use multi-ported RTUs.

Additional serial ports were added to RTUs, and separate protocol drivers were assigned to each port. Each SCADA system could interrogate the RTU (possibly using different protocols) on their port, and either could issue control commands to the RTU. A mechanism for coordination of who was or was not to issue controls was needed, and most systems with multiple ports had a scheme in which a parameter or flag in the RTU was used to identify the SCADA system that currently had control rights. The protocol software in the RTU had to be modified to examine

these flags when receiving a control command message, to determine whether to accept and execute the command. RTUs with multiple host connections were most common in the electric utility market. Dual (two) connections are the most common, but there have been situations with as many as five (5) separate SCADA hosts connected to a single, shared RTU.

In the common vernacular of SCADA systems, the RTU and host computer are said to be in a *master and slave* protocol arrangement. Today, a preferred replacement set of terms are *client and server*. This is particularly true with IP-based SCADA protocols. The master in this style of protocol is the computer that initiates all communications, and the slave responds to messages from the master. These two terms have decreased in popularity owing to their historical connotations, but it is still important to understand the concept. Today we might also choose to refer to this type of communication scheme as a poll-response architecture.

For polling and communications to work, one device must use the master (poll) message format of the protocol definition, and the other must follow the slave (response) message format. Normally, the communication ports on the RTU will be configured to support the slave portion of the specific protocol. However, in some SCADA applications, it has been impractical to provide direct communication channels to every RTU, and in those cases, selected RTUs have been used as a submaster or *concentrator.* In that case, these RTUs would appear as a slave to the SCADA host computer, but also appear as the master to RTUs for which it is acting as a submaster. In most cases the concentrator RTU would periodically and repeatedly poll all of its slave RTUs, independently of being polled by the SCADA host, and incorporate their data and status values into its database.

It is not unusual to have a large, higher-capacity RTU act as both slave and master, on separate communication ports. This concentrator RTU might have one port connected to a radio transceiver and be running a master protocol on that port, in order to poll several smaller RTUs in the general vicinity (fig. 2–4); its other port would run the slave protocol, to respond to host polling. In this sort of architecture, the concentrator RTU typically combines all of the inputs received from its slave RTUs into its own input table and presents all of this aggregated information to the host as a single, *virtual RTU*. For control outputs, the concentrator RTU identifies commands that are actually intended for control points in its slave RTUs, and it initiates the equivalent output control message sequence to the respective slave RTU. With a concentrator RTU arrangement, you also have the option of using different protocols on each of the ports, so that the concentrator is also acting as a protocol translator or converter. In truth, it is really functioning like a small SCADA master because polling of its own slave RTUs is normally done asynchronously to the polling for the data from those other RTUs by the central SCADA system. (The concentrator RTU polls its slaves and retains a local copy of their most recent data, which is what it uses to answer the central SCADA system's requests for such data.) Only with outgoing control commands, which must pass through the concentrator RTU, is there usually some form of coordination.

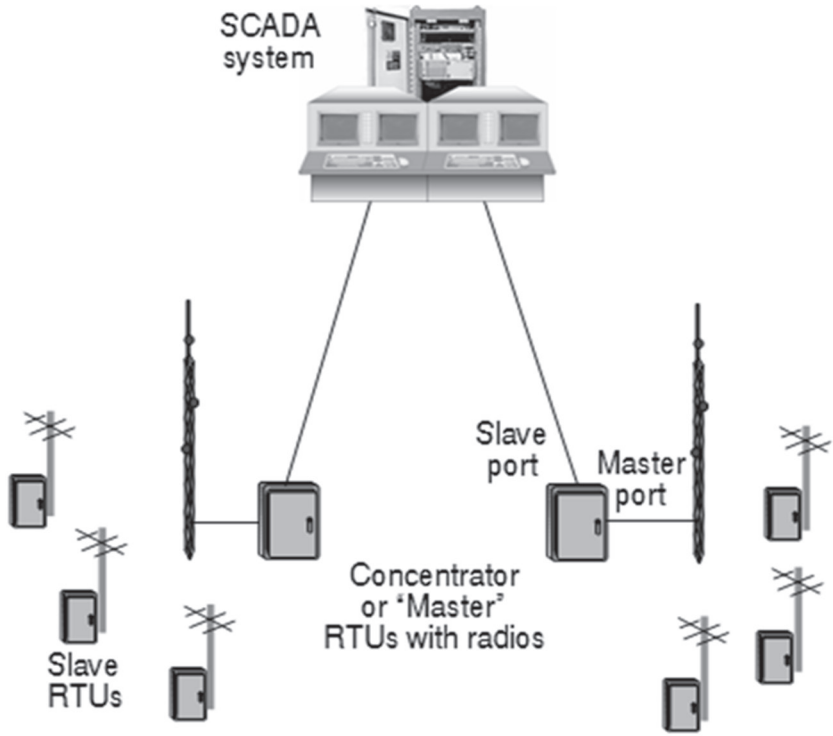


Fig. 2–4. RTU hierarchy using master and slave protocol combination

In recent years, especially in the electric utility market, it has become commonplace for RTUs to support a large number (16–32) of serial ports (while the amount of physical I/O has decreased). These ports are not for the RTU to communicate with the SCADA host, but to allow them to connect to and communicate with other nearby microprocessor-based devices. Many of the measurements and status points that would normally be wired to RTU analog and status inputs are being wired into other microprocessor-based smart devices (a.k.a., IEDs), and the values and state of these inputs can be obtained through serial communications with those other devices. The electric utility industry coined the term *intelligent electronic device* (IED) for such microprocessor-based equipment (well before that acronym started to have a very different and sinister meaning). *Substation automation,* or *substation data concentration,* is the process of using an RTU (or some other form of microprocessor-based device) with a large number of serial ports (and possibly no physical I/O at all) to collect all of the data from the IEDs in a substation, so that it can be sent to a SCADA system over a single communication circuit. Most RTUs manufactured today support optional multiple serial ports, and they can be configured as either a master or a slave port, using any number of available serial protocols.

Intercommunication is a problem in some cases, because of the complex types of data in the IEDs. RTUs were initially designed to support two basic data types: numeric values and binary values. Initially, the numeric values were integer only, although today most, if not all, RTUs can support floating-point or long integer values. Similarly, binary values were originally single-bit (two states) only; today multi-bit versions are supported (to represent devices with multiple states: e.g., a valve could be open, closed, or traveling).

Modern RTUs also support date/time information and quality tags that indicate the validity of the data to which they are attached. However, there are many other types of data, as well as data structures that combine multiple values and different data types. Many RTU protocols, even if they are IP-based, do not support advanced or sophisticated data types (e.g., multi-phase AC waveforms) and therefore may be incapable of extracting all but simple data from IEDs.

A common IED in the electric utility industry is the digital protective relay, the device that tells circuit breakers to open when bad things are happening on the power line. Inside a digital relay there can be dozens of measured and calculated numeric values and status bits. These relays also do SOE recording, power totalization over time intervals, and high-speed recording and capture of the AC voltage and current waveforms on all three phases and neutral. Getting this complex data by use of a conventional serial RTU protocol is, at best, very messy and is, at worst, impossible. Other types of protocols have been developed for this purpose (e.g., the utility communications architecture [UCA2.0] and DNP3), and the newest RTUs support such protocols and complex data types. Many electrical industry IEDs also support the DNP3 protocol, which is better designed than most, for extracting and delivering some of the more complex IED data types.

Local display

Since RTUs contain a lot of real-time process information, it is occasionally useful for a service technician or process engineer in the field to have the RTU provide a display of its information. This might be so that instruments can be calibrated or because no other display is provided for the data in question. Once RTUs became smart, it was possible to provide a range of electronic displays that could be driven and updated under software control. Once laptop PCs became ubiquitous, they could be used to make a local connection to the RTU, possibly via a serial console port connection. Today, an RTU might have an Ethernet port (or *Bluetooth* wireless capability) and could offer up dynamic Web pages on your laptop PC. However, in the early days of smart RTUs, such technology did not exist, so any display capability had to be built into the RTU hardware itself. An RTU might have a simple number/function keypad on its front panel and a multiline liquid crystal display (LCD) screen (fig. 2–5). The user could punch in an input number and see it displayed on the panel, or if no keypad was provided, the LCD data display might be set to cycle through all inputs on a continuous, repeating basis. If output

control was needed, then a keypad of some sort would be necessary. Importantly, the analog inputs and status inputs could be displayed only in an unconverted form (e.g., as *raw counts* or percentage of range), unless there was sufficient configuration data to allow a conversion calculation. Since building in a local HMI capability increases the manufacturing cost of an RTU and potentially adds a security risk, today it is more likely that an RTU would support Bluetooth or Wi-Fi and offer a Web-based local interface accessible via a laptop, tablet, or even a cell phone.

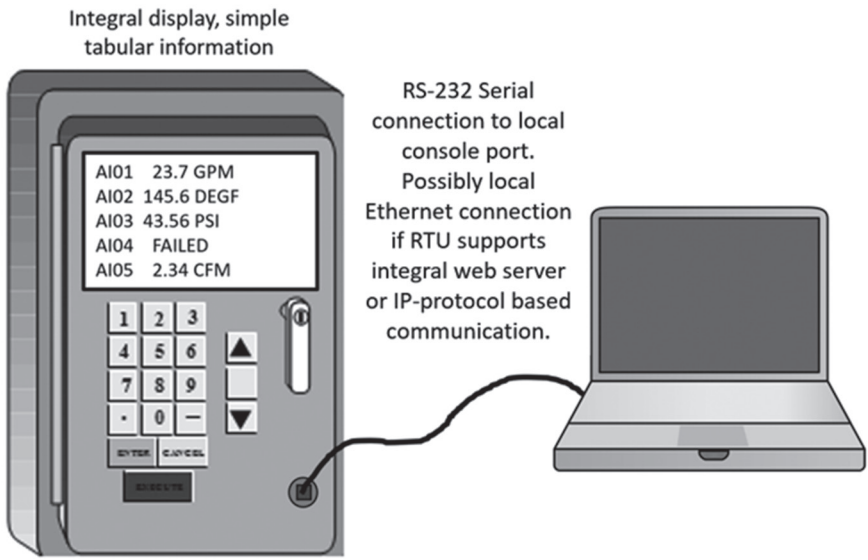


Fig. 2–5. Typical RTU multiline LCD and keypad

When an RTU reads an analog input, the A2D converter generates a binary number proportional to the range of the input value. Thus, if the input signal is 4 to 20 milliamps and the A2D converter is an 8 bit version, then at 12 milliamps (50 percent of the input range), the binary number from the converter should read "064_{10}" (i.e., half of the 000_{10} to 127_{10} conversion range). In this case, the RTU could present either the raw counts of "064_{10}" or the percentage of range this represents: namely, 50 percent. No additional data are needed to provide/compute these two display values. However, the input in question is actually a temperature, and the 4 to 20 milliamp signal range has been set for 150–350°F. What will be displayed for this input reading, at the host computer, will be a value of 250°F, which is the *engineering unit* value for the input. To compute this from the raw counts, the host has additional information about the engineering unit range; to supply engineering unit displays at the RTU, the RTU also needs a copy of that additional information. A similar situation occurs for status inputs. In the RTU, these will be seen as a 0 or 1 (or "on" or "off") raw value, but in the host computer, we know (because of additional data)

that a 0 means the pump is stopped and that a 1 means that the pump is running (and that is how the input would be displayed to the operator at the host computer).

With early smart RTUs, there was insufficient memory to hold tables of engineering unit conversion factors and status descriptors, and the protocols were inherited from dumb RTUs, which sent only raw counts and binary values. In the electric utility market, it remains common for RTUs to send all of their analog and status inputs as raw counts and for all engineering unit conversions to be done at the host level. Many of the protocols still in use in that segment support only raw counts. Modern smart RTUs are generally capable of the math required to convert raw counts to engineering units, but they do need the applicable linear conversion factors (m and b) for the Y = mX +b linear equation (for each input).

In the pipeline and water/wastewater industries, performance of calculations—and even sequential and regulatory control—is more common at the RTU level. Many of the calculations used are based on thermodynamics or physics and require that the equation elements be in defined engineering units: pounds per square inch, degrees Fahrenheit, cubic feet per minute, and so forth. Thus, it may be necessary to perform at least some engineering unit conversions within the RTU itself. In some RTUs, this is accomplished by user-written calculations or program logic in which the conversion factors were built into the logic or calculations. In other RTUs, the vendors have allowed the engineering unit conversion tables to be downloaded into the RTUs and have modified their communications protocols to support floating-point numbers. The DNP3.0 protocol handily supports floating-point values and multi-bit status value as well. It should be noted that when popular RTU serial protocols such as Modbus and DNP3.0 were turned into IP-based versions, none of the supported data types (from the serial versions) were augmented or altered.

An additional extension was the downloading of the alarm limit value tables, so that the engineering unit values could be checked against these limits. This required even more memory in the RTU and appropriate protocol extensions. RTUs that supported this full range of input processing were capable of providing local information displays that were much like those provided at the host (if equipped with the necessary display hardware). There were some instances of such RTUs having printers attached to them, so that they could print out alarm logs and reports. Today, RTUs (and PLCs) have extensive memory and computing capacity, and it is not uncommon to see engineering unit values produced in the field devices, and even given some level of alarm/quality checking and tagging.

Downloaded logic and parameters

The first smart RTUs were preprogrammed at the factory, and all of their functions were set into the firmware forever (to emulate the dumb RTUs they were replacing). But why have a computer and not make use of the added capabilities it offers? In the evolution of smart RTUs, there have been three levels of remote

configuration, each being successively more complex to implement but more flexible in application:

1. Preprogrammed functions with remotely adjustable parameters and flags
2. Downloadable library of user-selected and connected function blocks (much like DCS systems, smart instruments, and PLCs support)
3. User-written program/logic download and execution

Most RTU vendors provide one or more of these capabilities. With even moderately sophisticated PLCs, you have all of these capabilities. This could be through vendor proprietary functionality (e.g., support for C programs with a vendor-supplied function library) or by supporting industrial standards (e.g., supporting the IEC 61131 configuration standards).

It didn't take long for SCADA vendors to start adding simple function blocks and calculations that could be enabled and (re)parameterized via the communications channel. The logic and programming to perform the functions was still factory-installed, but the user could adjust *supervisory points* (binary and analog values stored in the software of the RTU) via the RTU protocol. Just as the RTUs supported real analog and binary I/O points, the smart RTUs added this new type of point, whose only purpose was to allow the host/operator to manipulate RTU pre-programmed functions by sending down supervisory point control commands. The idea of supervisory (software) points was eventually expanded to include both incoming points (to receive data and parameters from the host/operator) and outgoing points (so that RTU logic could send values to the host computer). Some obvious uses for supervisory points include parameter setting in simple logic and equations and changing details like meter factors in simple totalization calculations, the timing or duration of momentary contact outputs, and calibration scaling values. Using supervisory points eliminated the need to go to the field and change RTU firmware in order to make simple adjustments. Support for supervisory points required very minor enhancements to an RTU protocol. These points looked very much like conventional I/O except that they had no physical hardware associations. (This is a scheme used frequently in PLC applications, especially where Modbus protocol is used for the RTU/PLC communications.)

In addition to supervisory points, many SCADA vendors provided basic mechanisms, in the form of host-level configuration tools, for defining simple calculations and logic that could then be downloaded to the RTU for execution. Figure 2–6 shows two examples of simple calculations defined via a host utility. The text shown could be entered into a file via a text editor and then compiled (by another vendor-supplied application) into a data block that would subsequently be downloaded to the RTU, where a prewritten, pre-installed program would know how to interpret the data block and perform the defined calculations. Today, with RTUs (and PLCs that support IEC 61131), the configuration tools would allow *calculation blocks* to be defined and downloaded to the RTU (PLC) for execution.

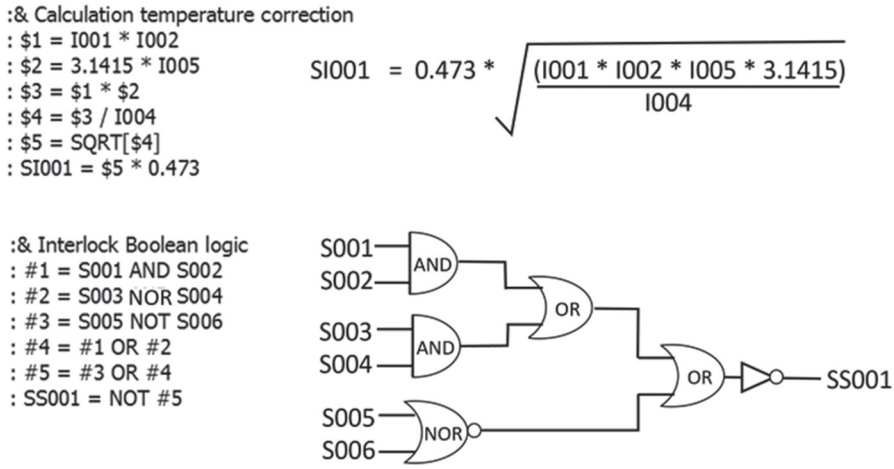


Fig. 2–6. Example host definition of downloaded calculation functions

In the examples shown in Figure 2–6, the "**$***number*" and "**#***number*" designations respectively are for numeric and binary temporary storage registers (in the RTU software) where intermediate values can be stored; the "**I***number*" and "**S***number*" designations respectively indicate physical analog and status input values are to be used in the calculation; and "**SS***number*" and "**SI***number*" designations respectively indicate that results are to be placed into supervisory status and analog points (where they can be fetched via RTU polling, just like real inputs). SCADA systems generally have a utility that permits the user to define any number of computations (using real inputs and the results of other calculations) at the host level. However, in the pipeline and water/wastewater industries, there are many instances where some level of calculation capability (using engineering unit values) is needed at the RTU level.

The addition of downloaded calculations and logic required further enhancements to protocols, so that there were message types for sending (and receiving) these data blocks and confirming successful delivery. If the calculations or logic incorporated supervisory points, this permitted additional flexibility, as such points could have their values manipulated through other RTU message types, as discussed previously in this section. Until very recently, these sorts of capabilities involved vendor-developed proprietary host-based utility programs that would input and process such logic and mathematical calculations, and convert them into data (or actual program code) that could then be downloaded to the specified RTU, either locally (e.g., via the console port) or over the polling circuit, using protocol extensions devised for such purposes. Within the RTUs would be software to deal with the data or install and execute the programming. Today, there are some widely adopted standards (such as IEC 61131) that have evolved for accomplishing the same results in a much more standardized manner.

Regulatory and sequence control

In the pipeline and water/wastewater industries, the RTUs installed in the field were often located at physical locations around the process where there was a need for some level of local regulatory control and sequence logic. Basic SCADA systems are *supervisory* control systems, meaning that the decisions to take a control action are made at the host level and then dispatched to the RTU to execute. This is fine as long as the communications bandwidth is sufficient and reliable. If gas pressure needs to be controlled at a delivery point and the target pressure needs to be adjustable based on conditions, then control adjustments to the pressure-regulating valve may need to be made every second or faster. With the low-baud-rate serial communication schemes used by most SCADA systems (not to mention channel sharing across multiple RTUs), it is typically not possible to read the gas pressure, send it to the host, make a control adjustment calculation in the host, and send a control output command to the RTU all within a second. In the past, in order to provide local regulation for such applications, it was typical to have some form of instrumentation and control panel at the site, to perform the regulatory control, and the RTU would merely interface with this panel via analog, pulse, and contact I/O points (fig. 2–7), typically to provide a reading of the measurements and to make setpoint adjustments to the instrument performing the *PID control*.

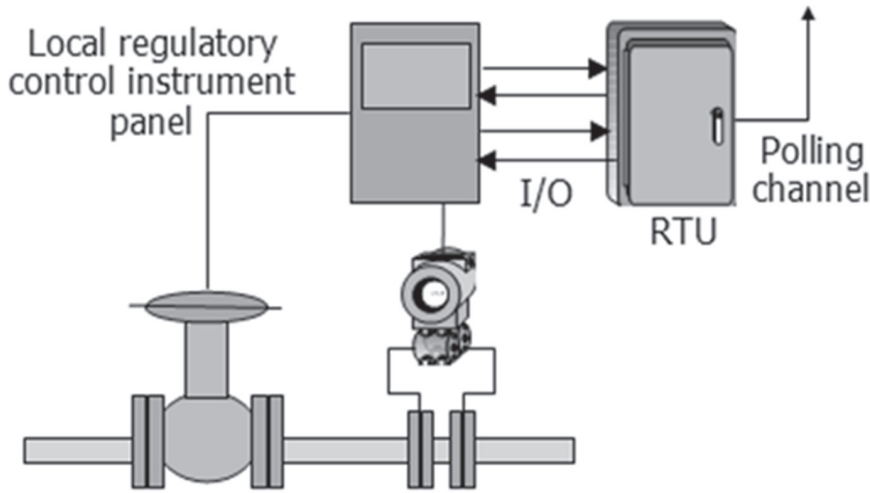


Fig. 2–7. Supervisory control of local regulatory control panels

In the 1980s, microprocessor-based RTUs had become reasonably powerful, and at the same time, a revolution was taking place in the process control industry. Traditional analog instrumentation and control panels were being replaced with computer-based technology. Specifically, the *distributed control system* (DCS) had been introduced and, for reasons both financial and technical, was

attaining wide acceptance as a replacement for conventional panel instruments and analog controls. A DCS consists of distributed process controllers with I/O, all tied together on a high-speed (for the 1980s) redundant LAN, or data highway, along with operator consoles and possibly other computers. The distributed process controllers were microprocessor-based and had a full complement of I/O. Architecturally, a DCS very much resembles a SCADA system, aside from the use of high-speed LAN communications instead of low-speed WAN communications (fig. 2–8).

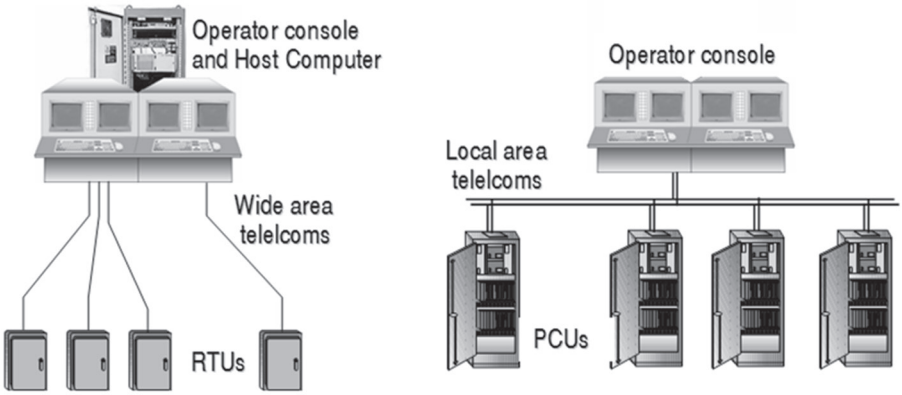


Fig. 2–8. Basic SCADA system and DCS architectures

There are many fundamental differences between these two types of systems, an important one being that distributed process controllers were intended to perform local regulatory and sequential control from the outset. The microprocessor technology and I/O technology of a process controller is not hugely different from that of an RTU, although process controllers usually came in fully redundant or one-for-*N* fault-tolerant designs, which would be very uncommon for RTUs.

It did not take SCADA vendors long to realize that they could add value to their RTUs by incorporating some of the control functions used in the process control industry. Computer-based process control systems had been around since the 1970s, although they used centralized, vendor-proprietary architectures (a.k.a. supervisory control). A set of algorithms had been developed by DCS vendors to mimic the actions of the analog controllers being replaced by DCS computer technology. The basis for most regulatory control functions was provided by algorithms such as the proportional-integral-derivative (PID) controller, the lead-lag algorithm, the ratio-bias controller, and the on-off ("bang-bang") controller. These were all numerical algorithms that could be either used individually or combined mathematically to process analog inputs and generate analog outputs, performing the analogous functions of the older analog panel instruments. SCADA system vendors, at least those serving the pipeline and water/wastewater industries,

started offering these algorithms in their RTUs (and PLCs). That eliminated the need for the local panel instruments, where customers were willing to accept an RTU as a process controller.

RTUs (and PLCs) have generally enjoyed a reputation for reliability. They are expected, after installation, to operate flawlessly 24/7 over the next 10 years or more. The fact of the matter is that most RTUs have done just that, and quite a bit more. (There are electric utilities today still using RTUs manufactured in the early 1980s.)

By contrast, many DCS vendors found that making a process controller fully redundant was just as difficult as the SCADA system vendors had found making their host computers redundant. Many DCS redundancy schemes failed to live up to their promise (just like the SCADA systems), and there were examples of process controller redundancy being disabled or removed by customers, to improve process controller reliability. At that point, a process controller and an RTU are very nearly the same device. Today many pipeline and water/wastewater SCADA systems still rely on the local regulatory control capabilities of their RTUs.

In addition to regulatory control (which generally involves analog inputs and outputs), many local control problems needed the capability of sequence and state-driven control, which is based on Boolean logic. Actually, a hybrid of regulatory control, altered and manipulated by Boolean logic, tends to serve best for sophisticated control. Many SCADA system vendors expanded their RTU features and functions to support both capabilities. The mechanisms used for this differed by vendor. Not all vendors could support top-down configuration (developed in the host and downloaded to the RTU), and not all vendors took a block approach. Several vendors developed their own special process control programming language (often a BASIC-like language), which included PID as a library function. Users developed sophisticated controls by writing programs in these languages and then loading these programs into the RTUs, possibly through the console port, prior to field installation. Today, support for the IEC 61131 standard is widespread and accomplishes the same results in a standardized manner.

IEC 61131 configuration

As various manufacturers of PLCs developed their products, each tended to develop their own programming and configuration tools. Early PLCs were limited to relay-ladder-logic, which mimicked the hard-wired logic the PLCs were replacing. Over time, as PLCs gained computing power and memory, it became possible to add more control functions into these products. By the mid 1990s, this had resulted with several major manufacturers selling incompatible products with proprietary programming/configuration tools. The IEC formed a committee and by the late 1990s, the IEC 61131 set of standards was released. Part 3 of these standards was the definition of five (5) methods for configuring the functions of a PLC, all of which could be intermixed as needed. These were the five:

- Relay Ladder Logic (a.k.a. ladder diagrams)
- Sequential Function Charts
- Function Block Diagrams (like DCS function block libraries)
- Structured Text (a lot like the Pascal programming language)
- Instruction Lists (like assembly language)

If a device complies with the standard, then any configuration tool from any vendor (not just the PLC manufacturer) that produces compatible output, can be used to generate a file that can be loaded into the device and executed. Depending on the device, that may require a local connection to the PC running the configuration tool, or it may be possible to download the file via a LAN/WAN connection. Figure 2–9 shows three examples of using the IEC standard: a simple ladder diagram, a sequential function chart, and a function block diagram. Many RTU manufacturers now support the IEC standard. From a cybersecurity perspective, it should be noted that there are many commercial packages that can be used to create IEC 61131 configuration files, so altering the logic and calculations in an RTU or PLC no longer requires access to vendor proprietary tools and utilities.

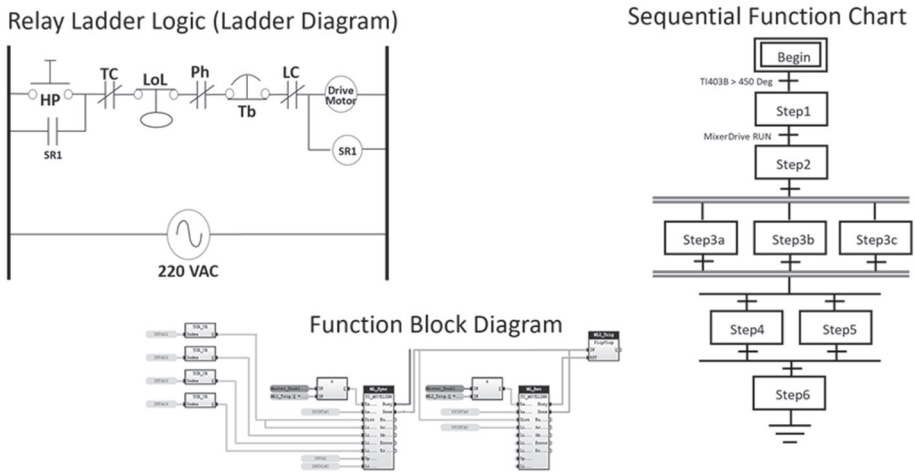


Fig. 2–9. IEC 61131 PLC/RTU configuration alternatives

Low-power operation

Another major difference between RTUs and process controllers is that process controllers are installed in industrial plants and manufacturing facilities where power is generally available, if not actually plentiful. In many instances, RTUs need to be remotely located at field sites that have no electrical power or telecommunications infrastructure. In those cases, the RTUs need to be capable of running on *solar power* (or other low-power sources, e.g., *running on thermoelectric generators*). An

entire class of RTUs has been developed through the years to operate at very low power levels (dozens of milliwatts), using schemes like powering off nearly everything until a polling message is received or powering up only for short time intervals on a scheduled, periodic basis. Many of the power-saving schemes employed in laptop PCs today were pioneered in low-power RTU designs in the 1980s and 1990s.

Low-powered RTU technology developments are primarily focused on the pipeline, transportation, and water industry segments (for remote locations such as a reservoir). This is generally because most water/wastewater RTUs and electric utility industry RTUs are near a source of electric power; don't forget, however, that when power goes out, that may be when an RTU is critical, and it will need to operate without the electric power that it was monitoring. There is a difference between very-low-power RTUs, which are capable of continuous operation from low-power sources, and conventional RTUs equipped with a *battery backup* facility (a.k.a. a UPS). A battery-backed conventional RTU is expected to run off a normal electric power source (the primary source), but it also incorporates a battery-based power supply that is maintained in a fully charged condition by the primary power source. A conventional RTU may require many watts of electrical power to function. On power loss, the RTU reverts to the battery-based power supply, either to ride through a short-duration outage in the primary power source or to take actions to shut down or bring the process to a safe/stable state. Battery-based power supplies support only a limited amount of operational time before the batteries are drained. With low-power RTUs, the power requirement of the device is generally reduced, through various means, to a sub-watt (milliwatt) level so they can operate from power sources such as solar panels.

Solar panels (which generate low-power ([milliwatts to a few watts]) direct current [DC] from sunlight) are a commonly used independent power source for low-powered RTUs because of the general availability of sunshine, at least during part of each day, everywhere in the world. (Along a gas pipeline, thermoelectric generators can be used, owing to the availability of gas as a fuel source.) A solar-panel power supply *also* has a battery-based backup component, to supply power during periods of darkness. This battery is normally recharged every day, from the power produced by the solar panels (which must also power the RTU), to make up for the power used during the hours of darkness.

Accumulator freeze

As previously mentioned, a common use of pulse inputs is to count them over time to determine material or energy quantities that have flowed past a particular measuring point. In pipelines (and electric power applications), this is an important task, because what goes *in,* should come back *out* elsewhere (adjusted for parameters like pressure and temperature). Pipelines generally count pulses as material (gas or liquid) moves past a measurement point. All inflows and outflows are metered in this manner. Thus, if we can capture the information about

what is currently passing in, out, and through the various measurement points in the pipeline, we can, at the same instance in time, figure out if there is a problem (e.g., a leak). If such measurements are not taken at the same time, then by the time that all accumulators have been collected, something more will have moved past the measurement point(s), and we won't be able to balance the inputs and the outputs. A big problem would be if we seemed to be losing or gaining material—indicating a leak or a miracle, respectively. Neither situation would be good.

The vast majority of RTU protocol messages are destined for a single, particular RTU, as designated by the RTU ID contained in the message. However, in many RTU applications, a few specific messages will need to be directed to all RTUs simultaneously. Such messages are called *broadcast message*s in protocol terminology. One common function of many RTUs, and a capability supported by their protocols, is the ability to send out an *accumulator freeze* broadcast message. The purpose of such a message is to get all RTUs to capture and save snapshots of their accumulators (pulse counters), at the same instance in time, and hold them for collection by the host computer. This becomes more complicated when there are multiple polling channels, as a mechanism is needed to stop all polling and then, once all polling channels are idle, to send out this freeze message on all of them at the same time. This is how it was accomplished with dumb RTUs (as well as on the first smart RTUs).

With smart RTUs, there was no reason not to allow each RTU to have a local *clock*. A clock is just a counter that, if properly designed, counts at an exact rate, so that time changes can be related to the count change. Early clock circuits were balky and temperature-sensitive and could drift off by a significant amount if not reset frequently. One broadcast message type supported by most smart RTUs and their associated protocols is a time broadcast that could be used to (re)set the local clock of all of the RTUs on the channel. Most SCADA systems broadcast the time from the host on a periodic basis, to keep RTU clocks in reasonable synchronization with the host clock. (Of course, in a modern IP-network-based SCADA system, this function is provided by having a [S]NTP server somewhere on the network.) This was an effective scheme when a time skew of many milliseconds was acceptable, either between the host and a given RTU or between numerous RTUs themselves.

Schemes that timed the message transit time from host to RTU were employed for improved accuracy. The *telephone company algorithm* ("At the tone, the time will be XX:XX:XX exactly") was used by many vendors. Their protocols included a broadcast message type that gave the time to all RTUs on that particular polling channel. But time lags in each communication circuit and in the message retransmission across all communication circuits still generally introduced small time skews. It was difficult, but possible, to obtain time synchronization, among all of the RTUs and the host computer, as accurate as ±100 milliseconds. For most applications, this was more than good enough. For freezing accumulators, it was generally acceptable. In those applications, the broadcast was not "freeze now"

but rather "freeze at time=XX:XX:XX" after having sent out time synchronization broadcasts. For the electric utility industry, though, it was insufficient to have time skews of many milliseconds. Electric utilities had achieved 1 millisecond SOE time-tagging accuracy *within* each substation on their major transmission lines, using the local RTU's clock or some other local time source. However, because of the problem with cross-RTU, cross-substation time synchronization, there was no way to correlate SOE data among substations, as the time tags couldn't be trusted to be on exactly the same time basis, owing to time skew between the substation RTUs.

Global Positioning System time receivers

It has already been mentioned that in some instances, the status inputs of an RTU will be used to time (or time-tag) events or to show the SOE. The local time tagging of a contact input state change merely means saving a record of what bits changed, to which state, at any given millisecond in time. To do this, an RTU must examine all of its status inputs every millisecond (or rely on hardware interrupts) and determine whether any have changed from their value at the prior millisecond. When changes are found, a record is made, showing the time (to the millisecond) and the inputs that have changed.

Even though RTUs have, for years, had sufficient computing power to provide a one-millisecond time scan of all status inputs, a problem remained in that the local timekeeping of every RTU could not be easily synchronized to the exact same millisecond. The only solution was to use a highly accurate time source that could be made available concurrently to all RTUs. For many years, the U.S. government has operated a radio station called WWV (there is another similar station serving the Hawaiian Islands and South Pacific region), whose purpose is like that of the time service of the telephone company. WWV broadcasts the exact time on a repeating, ongoing basis. Special receivers were available to listen to the WWV transmission and produce a time signal that could be fed into an RTU (often in the form of an *IRIG-B* signal). If all RTUs were so equipped, then they would all keep the same time, to a high level of precision. However, such receivers were expensive, and proper reception might entail erecting a tower and mounting a large antenna. Because of this, few utilities elected to avail themselves of this technology.

Then, in the mid-1990s, the U.S. government placed the Global Positioning System (GPS) satellite network into orbit, and suddenly we had a low-cost, high-precision time source, available as long as you have adequate visibility of the sky. GPS technology has made it possible to have RTUs anywhere in the world all be time-synchronized to one-millisecond (or better) accuracy. Furthermore, the SCADA host can use this technology too, so that the clocks in a SCADA system keep the exact time at all locations.

This brings up another idiosyncrasy of SCADA systems. Because they cover large geographic areas (e.g., with a pipeline that spans a good portion of the United

States), it is quite probable that the local time where one RTU is situated may be in a different time zone than another RTU and even the host(s). Rather than keeping local time in each RTU, most SCADA systems elect either to keep time based on where the operators (and host) are located (and provide RTUs a time adjustment factor) or to use Greenwich mean time (GMT) for the system time. This second choice may be taken to deal with another time-related problem: the daylight savings time adjustments made each autumn and spring.

Daylight savings time

Computer systems, and definitely SCADA systems, often store data chronologically (by date/time). The date/time information is stored in data tables. This causes a problem when, in the autumn and the spring, the time suddenly changes by an hour, plus or minus. This time-shift issue has plagued SCADA systems since they started collecting and archiving data. Imagine that you are recording a temperature and suddenly time jumps ahead one hour. It will appear that you had no data for an hour because the time stamp on the measurement recorded just prior to the change and the sample taken just after the time change will have time stamps an hour apart, even though they were taken within moments of each other. The opposite is even worse: if you have already stored an hour's worth of data time tagged as happening in hour X and suddenly it is hour X again, what do you do with the data you already recorded? To avoid such problems, many SCADA vendors use GMT in their systems (at least for internal time tagging of data) and never make those spring and autumn time adjustments. Others merely implement a time mapping function that figures out how to convert to and from local time and GMT time, accounting for such time adjustments.

Transducer-less AC inputs

The electric utility industry has a few RTU features and functions that are unique to that particular industry segment. High-resolution (one-millisecond) SOE recording has already been discussed. However, one of the other most important differences is that whereas to all other industries, AC 50/60-Hertz electric power is a source of noise to be filtered out of a measurement, to the electrical industry, that AC waveform is actually what they want to measure.

Until the mid-1990s, RTUs used in the electric utility industry had to employ external devices called *transducers* to convert AC voltage, current, power, and other attributes into a DC voltage or current value that could be read by a conventional A2D converter. By the 1990s, the digital signal processor (DSP) was finally powerful enough and cost-reduced enough to make it economically possible to use them in RTUs to take high-speed samples of the actual AC waveforms and then apply calculations, like the fast Fourier transform (FFT) or the discrete Fourier transform (DFT), to extract frequency-domain information. This also provided

a way to compute real and reactive values—and determine spectral component energy levels—for the electrical power signals being measured.

Today, transducer-less RTUs are common in the electric utility industry, particularly for 'pole-top' applications where space is at a premium. For the most part, few electric utilities have needed their RTUs to perform local autonomous control in their substations. That is the job of the protective relays, load tap-changers, and other such devices. But on distribution systems, an outage recovery scheme called sectionalizing has become a popular form of autonomous local control that can be implemented in RTUs. Getting electric power-specific values such as the power factor and frequency out on a transmission or distribution network was useful for controlling the grid, and transducer-less RTUs made it cost-effective to have more measurement points. RTUs in that industry have had few feature or functional enhancements (other than being forced to upgrade to newer microprocessor technology when older parts became unavailable). Nevertheless, over the past few years, the electric utilities have suddenly been looking at multi-serial-port RTUs as data concentrators for collecting and aggregating (simple) IED data in a single device that can then deliver this information to the SCADA system. (This trend, sometimes called *substation automation,* will be discussed in more detail in a later chapter.) Electric utilities are also making use of the cellular phone system in their feeder-automation and distribution automation SCADA efforts as generally cellular phone/data service will be available in those areas to which power is being distributed.

Top-Down and Bottom-Up Configuration

When SCADA systems were initially introduced, the companies offering them also designed and manufactured their own RTU equipment, including designing and implementing their own serial communication protocols and the operating software for their RTUs. In some instances, these companies were previously manufacturers of the non-computer-based telemetry systems that predated SCADA. In other instances, new companies were formed and others were acquired.

In the first 10 to 20 years of SCADA systems, most RTUs were purchased from a SCADA vendor. There were advantages to this arrangement. The SCADA vendors generally provided reasonably good tools (utility programs) for configuring the system, including the RTUs. With dumb RTUs, this amounted primarily to defining the number of RTUs on each polling channel and the I/O compliment of each. However, as smart RTUs replaced the dumb ones and as these RTUs were expanded with advanced capabilities, the configuration tools (and protocols) had to support these added functions. With RTUs that could execute user-defined calculations and logic, there had to be utility programs for defining these elements and for sending them down the communication link to the RTU.

Where RTUs were expanded to support user-written programs, the system had to support program development tools, as well as the utilities for sending

new program logic to the RTU. Certainly, there were SCADA system vendors that did not support remote downloading, although they did offer these advanced functions in their RTUs. In those instances, the vendors generally provided software tools that could be used to load these calculations, logic, and programming into the RTU, by a direct connection to the RTU prior to its being deployed and commissioned in the field, through a separate configuration port (possibly through the console port). Remember that personal computers (PCs) did not exist until the late 1970s and portable ("lug-able") and laptop PCs did not exist until the early 1980s. Thus, these software utilities were run on the SCADA system mainframe/minicomputer.

Although initially all RTUs were supplied by the respective SCADA vendors, as part of an overall system, SCADA vendors were not always able to maintain a successful and profitable business (many went out of business), and therefore in the mid to late 1980s, several third-party companies entered the field as RTU suppliers. There is a long list of names of former SCADA vendors that no longer exist, and most left behind customers with running systems and an installed base of thereafter unsupported RTUs. The owners of these systems had spent a great deal of time and money getting them installed and operational and training personnel in the use and maintenance of the system, making them disinclined to just throw the system away and find another vendor (who could just as well go out of business in the future). This situation created a business opportunity for the manufacture of third-party replacement or expansion RTUs that could be sold to the owners of these orphaned SCADA systems.

By the late 1980s, there were several suppliers that had entered the market and manufactured such RTUs. One of the challenges with utilizing a third-party RTU was that, even though the basic RTU protocol functions (e.g., reading inputs and controlling outputs) used by a SCADA vendor could be emulated or reproduced easily enough, there were advanced functions (for accepting downloaded calculations, logic, and programming) that could not be replicated. If you tried to use the SCADA system's configuration tools to send configuration files to these third-party RTUs, those files would typically be rejected. One reason for this was that the data and instructions generated by the SCADA system's tools would have been specifically structured and formatted for the type of CPU and programming logic (*machine code*) used in the SCADA system vendors' RTUs. Another was that even though the RTU vendors might have basic protocol specifications, they did not have the knowledge of the proprietary structure and content of the data files or program logic generated by the SCADA vendor's utility programs. These replacement RTUs therefore were limited to supporting the I/O functions of the RTUs they were replacing but not any of the remote configuration and logic downloading capabilities the old RTUs might have supported.

Complexity and cost made creating a viable emulation of the advanced functions and utilizing the host-created configuration files both technically daunting and financially questionable. Third-party suppliers were generally able to offer only

basic I/O polling and control functions. Since these RTU manufacturers could not or would not create the logic needed to deal with the advanced functions and those relevant protocol commands and data files, they had to provide some other way to configure their RTUs. Most used the RTUs' console ports and implemented configuration software tools that could be used to set up the RTU by connecting a PC (which by then existed) to that port.

Because of this inability to mimic the advanced functions supported by the downloadable RTUs from the original SCADA system vendors, the manufacturers of third-party RTUs had their greatest success in the electric utility market, where little or no use was made of such capabilities. Eventually (in the 1990s), RTUs became powerful enough to support *interpreted languages* (computer hardware-independent) and standards, such as the IEC61131-3 control logic standard that had emerged and gained wide support. Those factors, together with the emergence of standard protocols and *application program interfaces* (APIs), have created the current environment in which it is possible to mix and match SCADA software, PLCs, RTUs, and commercial application software to create a customized SCADA system.

The Emergence of PLCs

The biggest problem for many owners of SCADA systems was that vendors tended to sell them a system and then disappear, leaving them with an orphaned system and no ready source for more RTUs, software enhancements, and support. This happened a lot in the municipal water/wastewater industry because of its traditionally lowest bidder procurement practices—that is, selecting the lowest bidder and not the best-qualified supplier. (This has changed somewhat in recent years owing to the long list of never-completed or obsolete-when-delivered SCADA projects in that industry segment.)

At the same time that SCADA suppliers and third-party RTU manufacturers were disappearing from the scene, in the factory automation sector, manufacturers of PLCs were making considerable inroads by constantly enhancing the features of their products, improving the reliability of their products, and even reducing the cost of their products at the same time. PLCs had success in crossing over from the discrete manufacturing sector into the process control sector when they finally added analog I/O (initial PLC offerings had I/O suitable for machine control: discrete I/O, stepping motor controllers, etc.) and augmented their relay ladder-logic programming languages with function blocks that performed the regulatory control functions previously discussed in this chapter.

PLC manufacturers opened up their communication networks and protocols and invited third-party vendors to make complementary products. This attitude led to such developments as the Modbus protocol (introduced originally by AEG/Modicon) as a de facto serial protocol standard and numerous PC-based software

products for interacting with and programming these PLCs. The water/wastewater industries have two sides: water and sewage *treatment plants,* which are geographically constrained to a given facility and prone to using DCS or PLC automation technology, and *distribution or collection systems,* which are geographically distributed and often employ SCADA automation technology. Because of highly competitive pricing and relative ease of use, PLC equipment suppliers had a lot of success in getting their technology incorporated into water/wastewater processing and treatment plants. This eventually led to its use as RTU equipment in SCADA systems for these industries.

Today, just about every specification written and issued for the competitive public procurement of SCADA systems for that industry segment will specify PLCs, from one of the major manufacturers, for the RTU equipment to be included with the system. It is just as likely that they will require the Modbus protocol for the communications between the SCADA system and these field-based PLCs. PLCs today have extensive capacity for sequential controls, interlock/safety logic, regulatory control, and general calculations. Most provide programming tools that allow both local reconfiguration and possibly even remote reconfiguration via the communication protocol, depending on the protocol used.

Initially, PLC manufacturers developed their own tools and techniques for programming the logic to be performed by PLCs. Relay ladder logic (RLL) was popular because it mimicked the electrical wiring diagrams used by plant electricians for designing the hardwired relay logic that predated PLCs. In the 1990s, the IEC61131-3 standards were developed, and today, just about every PLC vendor and third-party software vendor complies with this standard. The standard defines a set of configuration/programming schemes that can be used to define the logic in a PLC (and today, even a fair number of RTU products). It includes not only RLL but also *sequential-function charts* (SFCs), structured English, function block chains, and other control logic and computational definition tools. Many DCS vendors now also offer IEC61131-3-compliant programming tools for the configuration of their process controllers. Many of the field sites where a water/wastewater RTU (now PLC) would be placed are locations where local regulatory control, interlock logic, and sequence logic were required. PLCs offer all of these capabilities. Most PLCs can also be easily equipped with a variety of off-the-shelf local operator panels, if required, because of the number of such devices that have Modbus protocol support.

Legacy Protocols

The SCADA industry has produced a huge number of serial protocols, both bit- and character-oriented, over the years. Their names or designations remind us of former SCADA system vendors: WESDAC, SGM, Tano-3, CDC Type- I, Tejas V, S3-type1, L&G 8979, and Harris 5000, among numerous others. In effect, all such

protocols do the same basic operations: send input values to a host computer and generate control outputs when commanded to do so by a host computer (fig. 2–10). Of course, beyond these basics, the different protocols tended to divide into classes or categories, based on the more advanced features and functions supported.

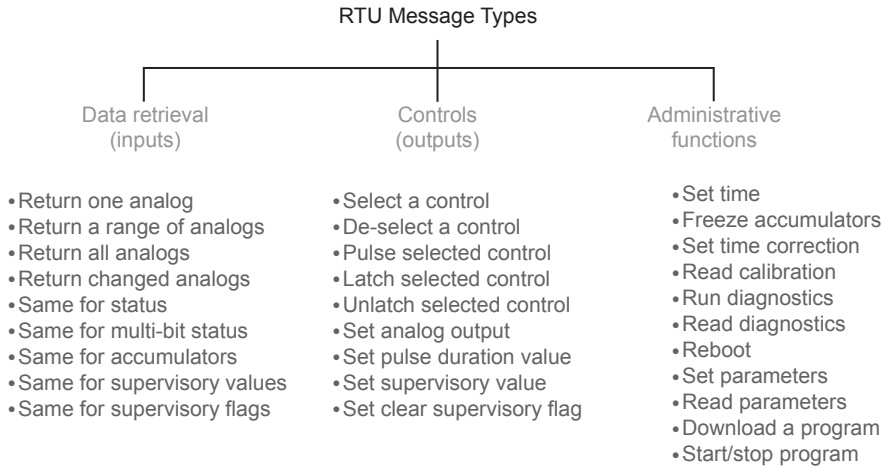


Fig. 2–10. Categories of typical RTU protocol message types

One very critical point about all such serial protocols (those intended to operate on slow speed serial channels with minimal overhead and limited, predefined functionality) is that they were never conceived or designed with any of the sorts of protections (e.g., encryption and authentication) that we can now employ for IP network-based protocols. This is due in part to the tremendous overhead imposed by such protective measures, the computing power required for such functions, and the very low data rates generally used for serial channel RTU polling. But it is also true that no one ever thought that SCADA systems and RTU communications would need to be protected. We know better today. One of the problems with legacy serial protocols is that there are limited commercial products available to secure those communication channels. A handful of manufacturers make "bump in the wire" link encryption devices that can provide a level of cybersecurity on a serial communication channel (wired or wireless). These devices can operate at serial data rate down to 1200 bps and up to 56 Kbps and accept RS-232/422/485 inputs. (Therefore, they can't easily be used with older RTUs that have a built-in modem circuit.) These link encryption devices need to be placed in the field device and at the SCADA system, and then all message traffic between the two devices is encrypted using 128 (or more) bit encryption. Some such devices can perform authentication as well, but most only perform encryption. It should

be noted that with a fixed-value encryption key and constantly repeated message traffic (e.g., poll-response), you greatly increase the likelihood of your encryption being broken. But link encryption is far better than no protection at all. Changing to IP-based networks and application-layer (OSI layer 7) industrial protocols potentially increases the risk of a communication compromise, but at least there are commercial technical solutions that can be used to add cybersecurity (confidentiality, authenticity, and integrity).

Most RTU serial protocols were designed for efficient performance over slow-speed communication circuits and thus avoided extraneous functionality that would increase message length. In the early days of RTUs (dumb ones), the MODEM technology available for leased telephone circuits offered a mere 110- or 300 baud (bits/second) rate. Shortly thereafter (in the late 1970s), the rates went up to 1,200 baud and then 2400 baud. That is what a lot of legacy SCADA systems still use today, even though MODEM technology has jumped well past those speeds. Many early RTUs had MODEM circuitry built into the RTU circuitry, so upgrading to a faster MODEM was not possible unless the RTU were replaced. If multiple such RTUs exist on a polling circuit, you must replace them all to upgrade the channel bit rate. (Although it has been tried, and there have been limited exceptions, normally you can't mix different protocols and/or data rates on the same communications channel.) Many of the older RTUs have very limited computing power and can't actually support high data rates, and some old SCADA systems have front-end processors that don't have the computing power or the serial port clock circuitry to handle higher data rates.

Many improvements have been implemented to make RTU protocols as efficient as possible once RTUs had adequate computing power (e.g., 16 bit CPUs introduced in the early 1980s), so as to reduce polling-response times or so that more RTUs can share a common channel without extending the overall polling time cycle. Early (dumb) RTUs generally did a *full report* of the current readings for all inputs (or possibly just all analog or all status inputs) every time they were polled. In many cases, these inputs had not changed since the previous polling request (especially the status inputs). Taking time to send the same values over and over again was inefficient and wasteful of time. With smart RTUs, it was possible to use the logic in the RTU to keep track of prior and current values for all inputs and to identify those that had changed. More advanced protocols (serial and IP-based) support *report-by-exception* polling, whereby the host doesn't ask for a report of values, but rather asks for a report of changed values since the prior poll request. A host could still ask for a full poll every so often (often called an *integrity poll*), to ensure that nothing has been missed. Successful exception reporting of analog values usually entailed some form of *deadband* setting that specified by how much an analog value must change (plus or minus) from its prior reading so as to require being reported as having changed. This is because analog values tend to drift around by a small amount, and thus an allowance had to be made to accommodate this or else all analog values would almost always look like they had changed since the prior poll.

The problem with polled exception reporting is that when all data aren't reported, additional information must be collected to identify which values are being reported as changed. There is a threshold above which this added information makes it better to do a full report rather an exception report. (For example, if all but one point has changed, the added data to identify all of the other values would be more bytes of data than reporting the one value that didn't change.)

A more recent advance in protocols is the development of *unsolicited report by exception*. In this scheme, the host computer does no polling (except an occasional integrity poll) and leaves it to the RTUs to send messages when values change. This scheme is best for processes where values change slowly and where you never see a lot of the measured parameters changing in lots of places at the same time

One problem with unsolicited report by exception is that you can't tell the difference between an RTU that has nothing to report (and so isn't sending any messages) and an RTU that has failed or lost communications. To differentiate between these possibilities, these schemes usually require either that the RTU send an occasional "I'm still alive" (a.k.a. heartbeat) message or that the host still polls every RTU occasionally.

Another problem with unsolicited exception reporting is *collision recovery,* which is when multiple RTUs decide to report their changes at the same time. This type of reporting is popular in applications in which a lot of RTUs must share a single, low-speed channel (e.g., a radio channel). But if multiple RTUs start transmitting at the same time, the host will just hear garbled noise. A mechanism is needed to ensure that collisions either don't occur or are detected and a process is followed to retransmit the data from the colliding RTUs. A mechanism used in some protocols is for the RTUs to send a *request to transmit* message (possibly several times) and only send the actual report once the host sends it a *permission to transmit* response, to ensure that only one RTU initiates message transmission at any given instance in time.

The primary mission of a communications protocol is to successfully and accurately deliver a message from one computer/device to another. Doing it efficiently and doing it securely are secondary issues. We have already discussed the incorporation, in most RTU serial protocols, of some means for identifying which RTU a message is for/from, in order to permit multi-dropping of RTUs on a common channel. All RTU protocols include some scheme that makes it possible for the message receiver to verify that the message arrived uncorrupted (by including something extra [a *check code*] with the message).

The robustness of these error detection schemes was greatly improved with the introduction of the cyclic redundancy check (CRC) code calculation in the 1980s. Prior to that time, a checksum and/or longitudinal redundancy check (LRC) was used for the purpose of detecting message corruption, but this proved to be untrustworthy (hence the use of select-check-operate control sequences). Several variations of the CRC calculation have been developed: an 8 bit, 16 bit and 32 bit

variation have all been used. Even Ethernet uses a CRC check (the 32 bit version) to validate message integrity. Unfortunately, beyond these features, all legacy serial protocols fall flat in regard to ensuring security and authenticity.

An RTU or host computer cannot actually determine who is sending a message on a multi-dropped channel. If a message arrives on the communication channel with the proper formatting and a valid check code, the RTU and/or host will accept the message as legitimate and process it accordingly. (More will be made of this in later chapters.) There are simpleminded protections: if the host sent a polling message to RTU X and got a reply from RTU Y, then it would probably discard the reply and repoll RTU X. But with off-the-shelf *protocol test set* software, it is incredibly easy to masquerade as a host or RTU and create valid-looking message traffic. There have been well-documented instances in which RTUs were commandeered by attackers using commercial radio communications equipment, a portable PC, and test set software. One of the first documented examples of industrial sabotage, caused by a cyberattack, occurred in Australia in 2000 in a network of sewage treatment plants.

Protocol Standards

Because of the number of incompatible, though essentially similar, serial protocols, there has been a push, from the SCADA system customer base, for a standard protocol that provides an all-encompassing set of features and functions, to replace all of the older proprietary protocols. The hope is to provide for compatibility and interoperability between and among vendors and products. Some popular proprietary serial protocols, like Modbus and BSAP (Bristol Standard Asynchronous/Synchronous Protocol), have become de facto standards in various industry segments because of their publication, ease of implementation, and widespread adoption. Others, like DNP3 (and also Modbus), were brought into the public domain and turned over to a committee for oversight and have been incorporated into a wide range of products.

With the rapid evolution of communications technology over the past two decades—particularly the development of high-bandwidth IP-based LANs and particularly WANs—other SCADA protocols suitable for use across IP-based networks have emerged. Protocols like IEC870-5-101 and -103, IEC61850, ICCP (Inter-Control Center Communications Protocol; also called TASE.2 or UCA1.0), and UCA2.0 were defined by committee, to run over LANs and/or WANs, and were then published and promoted by standards organizations. (Note that even these protocols did not initially incorporate sufficient [or any] security features, but all are currently being, or have been, reviewed with that objective in mind.) Popular serial protocols, such as Modbus and DNP3, have also be reimplemented as application-layer IP protocols and thus usable over IP-based networks. And has been mentioned, even if a given IP network-capable protocol doesn't

incorporate security measures, there are commercial software and hardware products (e.g., VPN gateways) that can be applied to provide that security.

Network versus serial protocols

A fundamental division separates currently existing protocols: there are serial protocols that are self-sufficient and only require a voice-grade (low-bandwidth, serial) communication circuit; and there are those protocols that expect a functioning digital network (LAN or WAN) of some type, composed of a communications circuit and an underlying set of basic networking protocols to provide *transport services*. When we discuss modern, digital communications protocols, we are generally referring to a class of protocols that would be considered as *Application Layer* (layer 7) protocols under the *ISO/OSI reference model*. Such protocols expect an underlying network to deal with intermediate transport of messages and ultimate source-to-destination message delivery. As an example, in a substation, if there are IEDs that are remotely accessible via the IP version of DNP3 protocol, these IEDs will most likely use Ethernet as the LAN connection mechanism to the communications interface device in the substation (e.g., a Router) and the interface device delivers messages to the wide area digital network (e.g., frame relay or a private IP network or even the Internet), which delivers them to another router and Ethernet LAN, where they get delivered to an engineer's desktop PC. Neither the IEDs nor the engineer's PC need to know about the underlying networking. They speak the IP version of DNP3 protocol to each other, and then the networking magic happens. With legacy serial SCADA protocols, we have single software modules (a.k.a., drivers) at each end that do all of the work and expect nothing between host and RTU except a reasonably reliable voice-grade communication circuit (fig. 2–11).

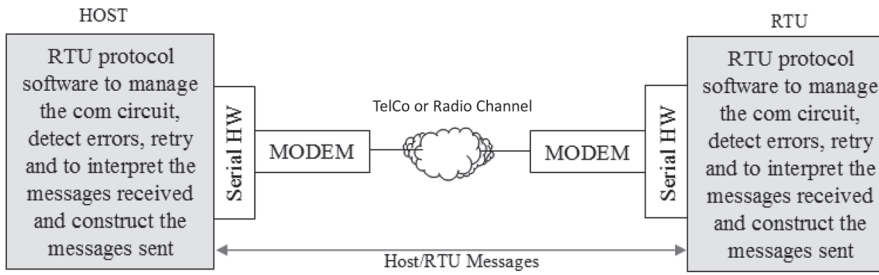


Fig. 2–11. Simple RTU serial protocol architecture

Some low-level digital network protocols can be used across the same voice-grade communication circuits that were used by traditional serial RTU protocols. For example, in a TCP/IP network, the lowest functional software layer of the protocol stack (the *Link Layer* or layer 2 in the OSI model), the one that deals with the message frames going in or out of the given computer/RTU, can be the serial

point-to-point protocol (PPP). In the bad old days when you dialed up over a telephone connection to your Internet service provider (ISP) to get onto the Internet, your PC was probably utilizing PPP (or the older SLIP) to move bits over the telephone line (using a MODEM as the electrical interface to the phone system). The problem is that there are layers above PPP (IP and TCP or UDP for example) that want to send their own messages (headers) to the other computer to manage the functions for which they are responsible (and consume bandwidth and time in this process). In addition, PPP carries a lot of overhead in its base message structure, to deal with more sophisticated communication issues not relevant to RTU polling. So even a simple analog input value report message will require many, many more total bits, if transported via a layered network delivery mechanism and protocol suite, than if delivered with a conventional serial RTU protocol (fig. 2–12). And since message length directly impacts polling speed, most serial protocols used for communicating with RTUs are stripped down to minimal functionality.

For these and other reasons (e.g., memory and computing power consumed by network protocol software), it was not generally viable, prior to the mid-1990s, to consider either using digital networking protocols over the slow, low-bandwidth communication circuits that were utilized by SCADA systems or running these protocols in older, 16 bit RTUs. Today, there are 32 and 64 bit processor-based RTUs with tremendous computing power, running commercial operating systems, and supporting communication speeds that, even over voice-grade circuits, are much higher owing to MODEM technology advances. Better still, telecommunication providers can supply *digital circuits,* rather than voice-grade communication circuits, and these offer the much higher bandwidth capacity (many tens of kilobits/second to multi-megabits/second) needed for real-time networking.

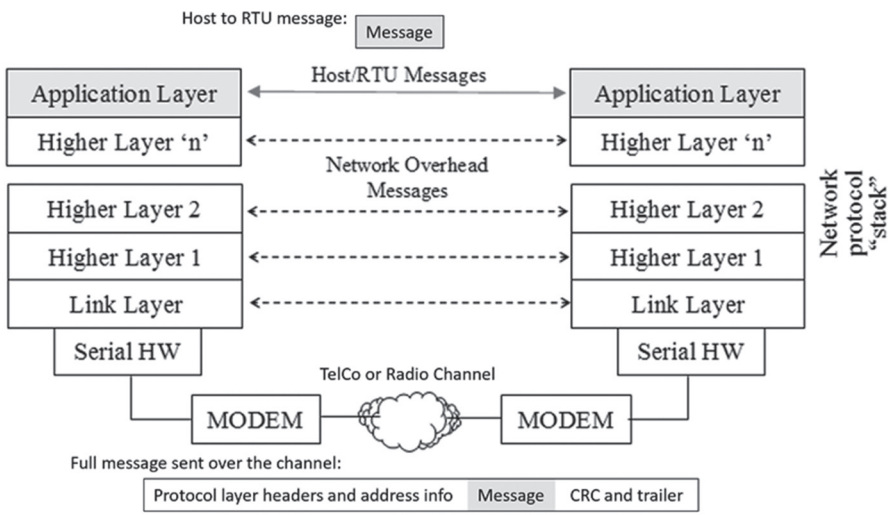


Fig. 2–12. Network-based serial protocol architecture

Encapsulated protocols

With the availability of IP-based LAN and WAN technology out to field sites, two well-accepted and widely adopted serial protocols (DNP3 and Modbus) needed a face-lift to make them available for use over such digital networks. Both protocols were re-released in the early 2000s, in an Internet Protocol (IP) version suitable for use over any network that was IP-based, regardless of the underlying physical network. (IP networking will be discussed in a later chapter.) This includes Ethernet-based local area networks and other wide-area networking technologies such as SONET, ATM, and Frame-Relay that can support TCP/IP as their upper-level protocol layers. Transmission Control Protocol (TCP) and UDP—part of the IP stack—are both commonly used to transport and deliver messages (application messages and data) over LANs and WANs. TCP and UDP don't really care what the data is; they are just responsible for delivering the data, preferably without errors. In this case, the application messages and data being delivered are the actual same Modbus and DNP3 commands and data that would have been sent over a serial communication channel, when the original serial versions of these protocols are employed. Rather than sending these bytes over a wire directly, they are delivered as the application data portion of TCP/IP (or UDP/IP) messages. (As a side note, in most of the instances where older serial protocols have been migrated into IP-Application layer versions, this process did not add to, enhance, or alter any of the basic functionality of the protocols.) The process of sending one type of message as data within another type of message is generally called *encapsulation*.

IP-Ready RTUs and Protocols

By the 2000s, the computing power of RTUs (and PLCs as well) had made a great leap forward, with the availability of lower-power-consumption (higher-temperature-range) versions of the then-popular Intel Pentium processors (and compatibles). At the same time, several of these newer RTUs ran full-featured commercial operating systems, with real-time extensions (such as Embedded Windows or a Linux variation). As a result, high-performance IP-ready RTUs were introduced that could support both legacy bit-oriented and standard character-oriented serial protocols, as well as having the LAN/WAN interface (Ethernet) and computing capability to deal with the more sophisticated, IP-network-based protocols. These RTUs came (come) with integral Ethernet and TCP/IP support and offer a platform on which a wide range of applications can be deployed. It must be mentioned that PLC technology was also following the same technical path and offered an alternative to these RTUs. These newer RTUs have been of interest to industries that were converting from traditional, analog serial communications and replacing them with some form of digital networking. This was also important because in that same time frame, most domestic telephone companies were moving away from supporting analog telephone lines (especially leased lines),

other than for residential connectivity. They were encouraging SCADA operators to convert to some form of digital network alternative (such as Frame-Relay).

One of the interesting features of this newer generation of RTUs is that because they supported extending IP networking to the field, they offered SCADA operators the possibility of employing the same network security technologies on these communication links as were being used to secure their corporate networks. Another interesting capability of these RTUs (if supported by the vendor and an appropriate IP-based protocol) is the ability to have *peer-to-peer* communications directly between and among RTUs, independent of the host computer(s). With most traditional serial protocols, the RTUs could communicate only with the host computer, so any inter-RTU data exchanges required host cooperation and support (DNP3 was one exception). One last aspect of these RTUs worth mentioning is their ability to support multi-session communications (which comes for free with a good TCP/IP implementation and sufficient memory and computing power). When a SCADA operator has replicated control centers, or just a fully redundant system design, most IP-ready RTUs can (given enough bandwidth) simultaneously process and respond to polling messages from multiple sources. This eliminates the need for one host (in a redundant pair) to poll all RTUs and send updates to the backup unit, or for messy communication switching between a primary and alternate operating site.

Currently, the list of IP-compatible RTU protocols is fairly short: DNP3, ICCP, UCA2.0, OPC-UA, and Modbus/TCP (plus those IEC protocols mentioned earlier). But there is really no reason to restrict oneself to SCADA protocols, as there are many fine and well-proven computer "IT" protocols that can be used for data transport (in fact, OPC-UA can and does make use of them). Certain limited SCADA applications use File Transfer Protocol (FTP) as a mechanism for data exchange between RTUs and the SCADA host. In other applications, *extensible markup language* (XML)-based Web pages have been used for those data exchanges, transported by the Hypertext Transfer Protocols (HTTP and HTTPS). And there are several IP-based so-called industrial protocols that can be used for host-to-RTU communications, such as OPC-UA, EtherNet/IP (with CIP messaging), and ProfiNET. This is particularly true where PLCs are used as the RTUs, as most PLC vendors support one or more IP-based industrial communication protocols.

If an RTU (and the host with which it is communicating) can support an IP-based protocol, the SCADA system owner has a wide range of commercial communication service providers and technologies available. (This will be discussed further in the next chapter.) This even includes utilizing the Internet itself as a communications system. A major domestic electric industry entity (an ISO) has already successfully demonstrated using the Internet as a means for real-time data collection from a large group of IP-ready RTUs located throughout a major Western state, protected by VPN technologies.

Another interesting aspect of TCP/IP networking is the ability to share a single communication channel across multiple applications. As will be discussed in a subsequent chapter, there are SCADA applications in which multiple host computers

need to access a common RTU. With conventional serial communications, this requires individual communication circuits from each host to the RTU site and an RTU with a sufficient number of serial ports. If digital (TCP/IP) networking is used, the RTU can concurrently handle communications transactions from all host computers, via its single connection into the digital network (much the same way that a Web site can service numerous browsers concurrently).

Some of the more advanced IP-ready RTUs and PLCs today are able to support an integral Web server function, so that the RTU can offer a range of Web pages to anyone with a laptop PC, a Web browser, and an Ethernet cable (fig. 2–13). Such Web page services may be used for data transfer purposes (e.g., using XML pages, as previously mentioned) or for human interaction with the RTU. Some vendors offer their bottom-up configuration functions via integral Web pages offered by the RTU. Others also provide diagnostics and real-time I/O value displays through these Web pages. Some vendors even offer tools that allow customers to create, or at least customize, their own RTU/PLC Web pages. It is not surprising that most of the IP-ready RTUs on the market today are running a Microsoft Windows or Unix/Linux operating system variation and taking advantage of open source Web server applications like Apache® or Abyss®, much like the commercial Web sites on the Internet. Some of these products are also implementing and taking advantage of the remote diagnostic and monitoring capabilities of Simple Network Management Protocol (SNMP). Of course, all of this opens up potentially exploitable cybersecurity vulnerabilities if not used properly. In a subsequent chapter, we will discuss and address various approaches for ensuring SCADA system cybersecurity. But with RTUs sunning a COTS operating system they need to be considered as potentially as vulnerable as a PC and thus protective measures, such as hardening and firewalls, may need to be applied to those RTUs as well.

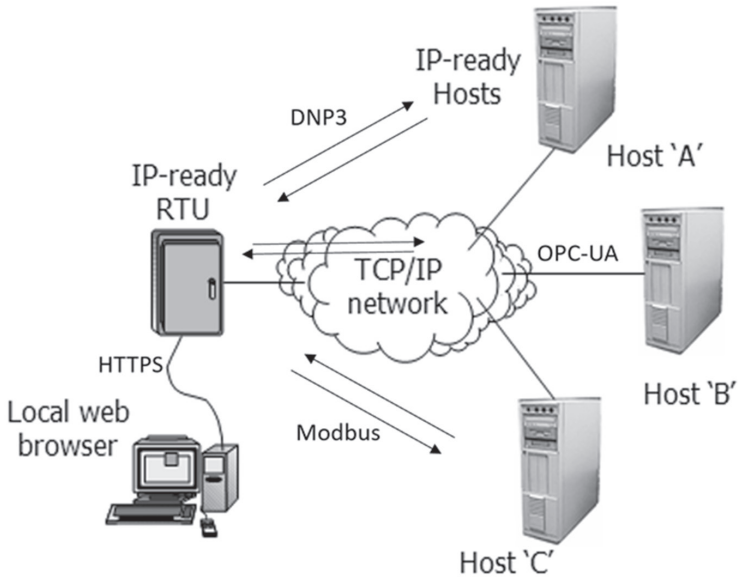


Fig. 2–13. Ethernet cable

3

Telecommunications technologies

A basic fact of SCADA systems is that they need to utilize communication technologies that can provide connectivity, and adequate performance and reliability, between the control center(s) and all of the remote field sites. The available technologies that could address those requirements have changed quite a bit since SCADA technology emerged in the 1960s. Today, we not only have a range of wired and wireless digital communication options, we have the ubiquitous Internet, which reaches nearly every populated area of the world (even out to the International Space Station). Since the communication system is one of the potential attack pathways that could be used for a cyberattack on a SCADA system, we need to understand the good and bad points of the various technologies. Also, since this book tries to address existing, legacy SCADA systems as well as new(er) ones, we need to cover legacy communication technologies that may still be in use today, since in the world of industrial automation, if stuff still works and can be maintained, we tend to keep it going for as long as possible.

Voice-Grade (Analog) Telephony

When SCADA systems emerged in the 1960s, there were only two choices for long-distance communications: the telephone system (or telephone system technology), built and operated by the yet to be broken up Bell Telephone Company ("Ma Bell"); and private, licensed radio. Telex and telegraph still existed but were not worth consideration for SCADA applications. Both of these technologies (licensed radio and telephone) were designed for the long-distance, bidirectional transmission of voice/sound (although it was also possible and common to send *Morse code* over a radio channel). Most of the communications technologies we take for granted today—satellites, cell phones, and the Internet, among others—did not exist (although the first satellite-based telephone links, between Europe and the United States, were established in the 1960s). Thus, in designing SCADA systems—and in particular the mechanisms for transporting messages between the field-based RTUs and the SCADA host—an approach compatible with these two available technologies was required. If you ask the question "what is the basic business of the telephone company?" the fundamental answer is "to move

a reasonable facsimile of the human voice from point A to point B continuously, in real time." The telephone company figured out that this takes a continuous stream of approximately 64 kilobits of data per second (in the digital world) to deliver speech from one point to another, and that particular number forms the basis for most telephone company bandwidth metrics. (You will see a circuit with this bandwidth called a DS0 or one voice-grade channel. Other, higher-bandwidth channels—e.g., T1/T3—are rated in terms of how many DS0s they support. As an example, a 1.536 Mbps (million bits per second) T1 circuit can be decomposed into 24 DS0 voice-grade channels.) In the late 1960s, the phone company was already experimenting with digitized voice and computer-based message routing (what they called *digital switching*). And today, that is the basis for all telephone service technology, especially cell phones and VoIP phones.

The same function (delivering voice from one point to another) is essentially true for radio communications (unless you are a 'Ham' [amateur] radio operator and like sending Morse code). SCADA systems don't need to carry on a conversation between the SCADA host and the RTUs; they need to exchange binary numbers that represent data and commands and to do this without introducing errors or delays. In the 1960s, there were fledgling remote computer communications systems in place. Computer time-sharing systems existed, and MODEMs were used to convert electrical, binary-digit (0 and 1) signals into sounds that could traverse a telephone or radio channel. Even in the early 2000s, with MODEMs capable of 56 kilobits per second over an analog telephone channel, the actual data encoding was still in the form of sound. The squealing sound that you have heard when you dialed, through the telephone system, into your ISP (if you're old enough to remember that and to have used AOL to reach the Internet) is the sound of a MODEM sending 0s and 1s (although with modern MODEMs, it is a bit more complicated than simple tones and individual bits).

Telephone technology

The (analog) telephone system of the 1960s did not add value. (Today, that is not the case. You can have call forwarding, voice mail, caller ID, etc., as part of your service.) The telephone company provided a reasonably reliable, somewhat noisy, low-bandwidth (voice-grade) channel over which you were welcome to send analog electronic signals in the frequency range of the average human voice. You could generate those signals with the telephone handset they provided (and by speaking) or with some other electronic device, such as a MODEM. (The earliest modems actually used an 'acoustic-coupler' which was a set of rubber cups that fit over the two ends of the phone handset, one with a speaker and one with a microphone). Establishing a connection (dialing), maintaining a connection, selecting a signaling mechanism and symbol set, detecting and correcting errors, managing and pacing the data flow, and terminating the connection were all considered to be the responsibility of those who wanted to use the telephone system. (In a later section dealing

with network-based IP protocols, these functions will be similar to those provided by TCP/IP.) For most consumers, the establishment of a call began with dialing a telephone number and waiting for someone to answer. The process of dialing takes a surprising amount of time, and this was even more time-consuming back when pulse/rotary dialing (versus touch-tone) was still used. For a SCADA system to have to dial a telephone number to reach an RTU (and then do it again and again) would thus be inefficient.

For best efficiency, a SCADA system needs a permanent telephone connection to its various RTUs. The telephone company accommodated SCADA system owners by providing private (analog) *leased lines*. These were telephone circuit connections that had the same appearance and functioned as if a number had been dialed and answered, but the circuit was locked such that it never disconnected or broke any of the intervening connections. Actually, with older electromechanical telephone switching equipment, this was implemented by going to the various switches and placing actual metal clips on the connections so that they were mechanically latched. That is not to say that the telephone company was providing a *metallic circuit* between the SCADA host and the RTU; the leased line was a voice-grade circuit with amplifiers and coupling transformers along the way to boost the signal, not just a set of wires (and eventually even fiber-optic cables).

These leased circuits—and the telephone system in general—were *analog* in the 1960s (and well into the 1970s). This means that you could connect a telephone MODEM (or even a handset) to the telephone line at any point along the leased circuit and eavesdrop on the SCADA-RTU messages going back and forth (or even inject some false messages).

Today, some (but less every year) SCADA systems still depend on leased analog phone lines provided by the local telephone company, although the underlying technology of the telephone system is no longer analog. Starting in the 1970s and finishing in the late 1980s, the telephone system was converted from analog to digital. Only the *local loop,* from the nearest telephone switching center to your home, is analog anymore (for those of you who actually still have "land lines"). Everywhere in between is digital (The same is true today for those of you that receive phone service via your TV cable company). For a SCADA system that uses analog, leased telephone circuits for RTU communications, the local loop at both ends of each circuit is a potential security vulnerability. With readily available equipment, it would be possible to record, replay, or simulate valid-appearing message traffic at either of the two end points.

A typical business office will have its own private branch exchange (PBX) that deals with analog telephones but makes a digital connection (via one or more T1/T3s) into the public telephone system. For SCADA systems that still use analog leased telephone lines, this often means that the telephone system is digital up to the panel in the telecommunications room of the facility where the SCADA host is located—and analog from there up to the room where the SCADA system front end and MODEMs are physically placed. In the field, this means that the circuit is digital to the nearest switching center and analog to the field RTU site.

For electric utility substations, many telephone companies have a policy of running an analog telephone line to the *demarcation point* outside the substation facility (usually a well-marked and obvious box) and making it the responsibility of the utility to bring the line into the substation and to the RTU. This means that the analog (and easily tapped) leased telephone circuit exists only at each end of the line (but often in unprotected areas).

The telephone system provides a *full-duplex* communication channel: sound can be sent in both directions at the same time—unlike *half-duplex* channels, as provided by a walkie-talkie or an intercom, and *simplex* channels, such as television or a public address (PA) system, which only send information in one direction. In most simple, serial SCADA protocols, that full-duplex capability is not utilized. The vast majority of legacy serial protocols work totally on a half-duplex basis: a message goes out from the SCADA host to the RTUs, and the appropriate RTU formulates a response and sends it back. (This is called a *poll-response* scheme.) This isn't an efficient use of the full-duplex nature of the telephone system, and modern computer networking protocols usually support data flowing in both directions at once. One reason that SCADA serial protocols don't utilize this full-duplex capability is that a protocol needs to be far more sophisticated to manage two independent, concurrent data streams. The other reason is that most such protocols, when they were developed, needed to work over the other major available communications technology: radio, which tends to be a half-duplex channel. In fact, radio can be used only in a full-duplex mode with point-to-point applications (such as between your cell phone and the cell tower) and not with point-to-multipoint applications. (For full-duplex operation, you must transmit on one frequency while receiving on a different frequency, and the other party must be setup reversed. This works well between two parties but is useless among a larger group that needs to communicate with each other.)

Analog (leased) telephone circuits are far less commonly used today, and any SCADA operators still needing them are finding that they are either paying a premium for, or are unable to obtain, additional analog circuits—and that their telecommunication provider(s) is pushing them to transition onto digital circuits for data services. The telephone companies don't want to maintain analog circuits (which are really digital up to the end points), but a large percentage of legacy installed RTUs and SCADA hosts are incapable of directly connecting to a digital network. Some very old RTUs still in use even have built-in analog MODEMs (usually a Bell 202 series), which in transitioning to digital networks usually makes RTU replacement a requirement. If there are multiple RTUs on a given leased telephone line, you can't change communications protocols or the data transmission rates on that circuit unless you can simultaneously change it for all of the RTUs (or take the circuit and RTUs out of service till they can all be replaced). Although analog telephone lines restrict data transmission rates to low speeds (generally under 34 kilobits per second [Kbps]), they have been sufficient for basic SCADA applications, because traditional SCADA systems and RTU protocols were designed

for even lower transmission rates (1.2–9.6 Kbps). One approach to addressing a transition to digital networking is to utilize *terminal server* technology. A terminal server (also called a *console server* or *serial gateway*) is a device that accepts an Ethernet-TCP/IP connection and provides one or more RS-232 (or RS-422/RS-485) serial ports for local connection to devices that require this. Some terminal server devices can even be equipped with a MODEM output. Special driver software is used in the host computer to generate virtual "COM": ports through which the SCADA software can poll the RTUs connected to these terminal servers. This software, communicating with the terminal servers, provides the equivalent of a dedicated serial communication link. Some legacy SCADA systems have been able to be adapted to digital networks by using this technology. (It usually required recompiling at least some of your SCADA software, so unless you have the source code and necessary compiler/linker utilities, you won't be able to utilize this strategy.) Unlike actual leased serial/analog circuits, you use an IP-based LAN or WAN connection to connect to terminal servers, which means that security mechanisms such as VNP (which we will discuss in a later chapter) can be applied to protect and secure these communications.

Licensed radio

Radio communications technology from the 1960s falls into two categories: point-to-point, high-bandwidth microwave transmission, and point-to-multipoint ultrahigh-frequency (UHF) and very-high-frequency (VHF) radio transmission. The telephone company pioneered the use of microwave transmissions to carry a great number of *multiplexed* communication channels over long distances. You still see microwave repeater towers in the countryside in some parts of the United States, although most of that infrastructure has been demolished (or abandoned) and replaced by fiber-optic networks. For some vital, geographically disbursed applications that use SCADA technology (e.g., power transmission), the owners/operators have had to make a decision about relying on third parties (e.g., the telephone company) for critical SCADA communications, or building and operating it themselves.

In many instances, electric utilities and water companies, whose service areas were covered by a local telephone company, may still have elected to construct and maintain their own communications infrastructure, rather than being dependent on a third party or as a backup. Water utilities have often chosen to implement a UHF or VHF radio–based SCADA communications system, with one or more master radios used to send/receive messages to RTUs equipped with radios operating on the same frequency. To cover more area or to speed up polling times, multiple master radios, operating on different frequency channels, could be used, each with its own set of RTUs with radios set to their particular master radio's frequency (fig. 3–1). In this case, the various master radios might be remotely located from the SCADA system and connected over leased phone circuits. One

of the challenges with UHF/VHF radio is that anyone with a radio set to the same frequency, and an RTU test set, could potentially spoof message traffic between the SCADA host and the RTUs, particularly since the protocols being used generally have no security mechanisms. There have been documented instances of such command spoofing cyberattacks on radio-linked industrial facilities. Adding security/encryption to low-speed, serial channels usually requires applying external devices, such as link encryption units, because the SCADA protocols themselves incorporated no security mechanisms.

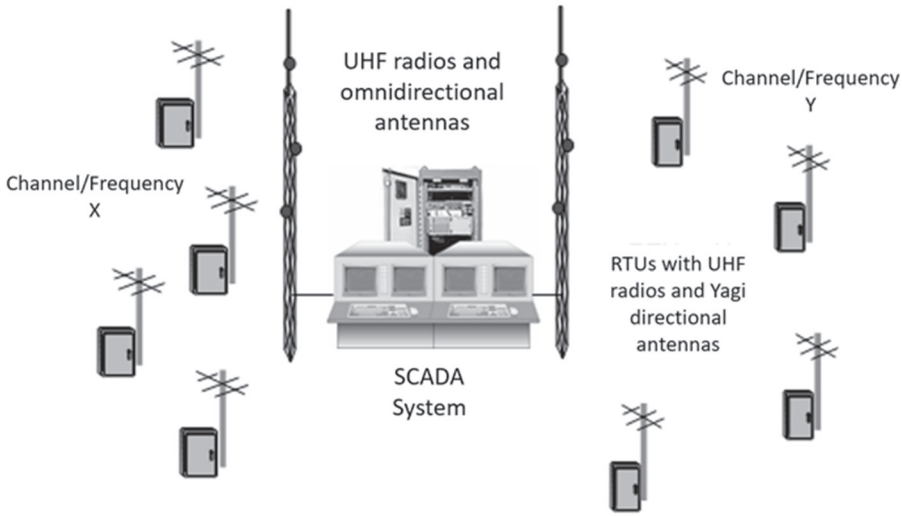


Fig. 3–1. SCADA host with multiple master radios on separate frequencies

A potential problem faced by SCADA system operators trying to use licensed radio is the congestion on the available frequency slots, particularly in urban areas. It is not always possible to find an available, open frequency slot. The Federal Communications Commission maintains a geographic directory of frequency assignments, so that overlap and interference can be avoided (a handy source of information for someone who wants to know the radio frequency you use to communicate with your RTUs).

Just as RTUs became intelligent with the infusion of microprocessor technology into their designs, radio equipment also became smarter. Originally, radios were just a transmitter and receiver that were controlled by the SCADA host or RTU, just like a person controls a walkie-talkie. A MODEM circuit was used to convert the binary digits into sounds, and the RTUs and SCADA hosts had program logic to deal with keying the radio (switching from receive to transmit mode) and providing time delays between the end of a receiving a message and starting the transmission of a reply. It was also necessary for every RTU to listen to (and

interpret) every host message, so that it could check the RTU ID in each message and find out if it was the recipient of that message.

With the integration of microprocessors into radios, the "digital" radios could use a small portion of their bandwidth to send messages to each other (a process called *out-of-band signaling*), and the radios could make decisions and identify messages that the RTU actually needed to hear. Radio systems became a bit more flexible with the introduction of the multiple address system (MAS) and trunked radio systems. The connection between the radio and the computer became a serial (RS-232) connection (i.e., computer to computer), and the radios could identify that messages were being directed to them (the radios had a unique ID); thus, RTUs received and had to process only those messages sent specifically to them by the host. Radios could also be configured as relay (repeater) devices to both extend range and to get around obstructions. Today, we have a multitude of unlicensed wireless technologies that can be used to cover a moderate service area, including radio repeaters and mesh radio networks. Much of the more modern radio equipment also incorporates security mechanisms such as 'strong' authentication and encryption.

Broadcast radio propagates outward from the transmitter in a circular pattern, like ripples in a pond when you toss in a rock (assuming you use an omnidirectional antenna). The circular radius may extend as far as 20 to 30 miles, based on terrain. That allows a large geographic area to be covered by the signal, and it can be sufficient to blanket a large urban area. Notably, conventional UHF radio technology provides a low-bandwidth (voice-grade) channel and offers no intrinsic security features. Also, broadcast radio has line-of-sight limitations unless *repeaters* are employed to extend the range by retransmitting (boosting) the signal and to bypass such physical barriers as hills and tall buildings.

Communications backup

Although utilities might elect to construct, operate, and maintain their own private radio communications systems, they could also use the telephone system as a backup in case of radio failures. It was not uncommon, once multi-ported RTUs became available, to have critical RTUs configured with an auto-answer MODEM and dial-up telephone line, as a backup to the primary radio polling channel. If the SCADA host could not reach a critical RTU via the radio system, it could dial a telephone number and reconnect to the RTU via the public telephone system, to poll the RTU or to send a control command. If there were multiple critical RTUs and none could be reached via radio, the SCADA host could cycle through the set of RTUs by dialing them consecutively, polling, and then hanging up and dialing the next in sequence. This wasn't very fast, but it was much better than having no communications at all. The opposite strategy has also been frequently used: use leased telephone lines as the primary polling and communication mechanism, and a fall back to radio to reach the critical RTUs if the telephone system goes down.

Of course, today's SCADA system operators can also consider the use of cellular telephone technologies, rather than conventional land lines.

Private telephone systems

Although it could be very convenient to call the telephone company and ask them to supply a leased telephone line from point A to points B,C, D, etc., there was a problem if those various points were not in the service area of the local telephone company. For processes covering great distances and located away from civilization, often the only practical choice was to construct a private telecommunications system along the pipeline (railroad track, power transmission line, highway) *right-of-way*. In those instances, the typical approach was to use the same technology and equipment that the telephone company used: constructing a microwave repeater station backbone along the right-of-way and using telephone multiplexing equipment to create a voice-grade telephone system. Such systems would have the capacity for a large number of voice-grade channels, and a relatively small number of those channels would be allocated to the SCADA host polling functions, with the rest used to provide telephone service to the remote sites (fig. 3–2). The SCADA system owner would purchase telephone equipment to build their own communications infrastructure in the same way the telephone company would have, had it been willing.

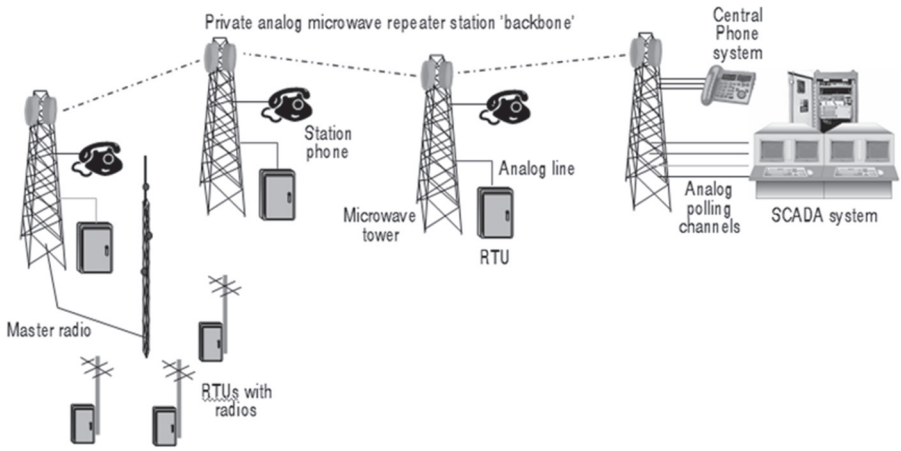


Fig. 3–2. Typical microwave-based private telephone system

As telephone technology evolved to include satellite links and fiber-optic cables, the organizations that elected to construct their own telecommunications infrastructure tended to migrate to those technologies. Many owners/operators of SCADA systems (particularly electric utilities and pipeline companies) ended up creating large and highly competent telecommunications support organizations to

maintain and extend their private telecommunications systems. Unfortunately, as technology shifted to digital networks and IP-networking in the mid-1990s, few organizations expended the funding or effort required to upgrade the skill sets of these internal telecommunications groups. Today, as IP-based digital networks become the main communications technology for SCADA systems, corporate IT organizations have tended to be given responsibility for supporting these networks.

Private telecommunications systems continued to evolve as the telephone companies and Bell Labs (and its successors, the "Baby Bells") introduced new technologies like fiber optics and synchronous optical networking (SONET). By adopting these technologies, many SCADA system owners both reduced the initial costs of their private telecommunications infrastructure and also ended up with excess bandwidth capacity that they could sell or lease to the telephone companies or regional/local municipalities. Electric utilities were stringing fiber-optic cable on their transmission lines, and pipeline companies were burying fiber-optic cables alongside their pipelines. In fact, many utilities entered the telecommunications market by selling or leasing their excess communication bandwidth or *dark fiber*. In a couple of surprising instances, electric utilities formed separate business units to market this excess bandwidth and ended up not retaining enough capacity for their own SCADA system. Go figure.

Commercial Voice/Data Carriers

In the 1960s, organizations that preferred not to construct, operate, and maintain private telecommunications infrastructure had few alternatives if Ma Bell couldn't meet their needs. But their choices started to expand in the late 1970s, and today, there are numerous options. In the 1970s, there was a major adoption of computer technology in the commercial and retail world. Point-of-sale terminals and smart cash registers replaced prior technology. Stores linked their branches with headquarters for faster inventory and accounting updates. Systems like the old American Airlines SABRE (Semi-Automated Business Research Environment) airline reservation system were expanding their coverage. All of this meant a growing need for computer communications. The telephone company offered a poor set of overpriced options to a company that had numerous branches full of dumb terminals that needed to send occasional data bursts back to the home office. You could lease a high-bandwidth, high-priced T1 circuit to each operating branch and have it hardly utilized (based on average bandwidth usage)—and then do that for every branch. Commercial data carriers came into existence to compete for these dollars.

X.25 packet switching networks

To better address the market's need to move lots of small data bursts from lots of places to a few central locations without spending a fortune on communication circuits, independent commercial data carriers sprang into existence and devised a new technology. These commercial data carriers were companies that placed computers around the United States in major cities and then leased those expensive T1 and T3 circuits from the telephone company. They would drop a packet assembler/disassembler (PAD) at a company's various facilities and headquarters (and the facilities of other similar organizations in the vicinity), and that company would pay them for the messages they sent, not for any fixed amount of bandwidth. By getting a sufficient customer base in an area, they could keep the leased T1s full, yet charge customers for just the fraction of bandwidth that each actually utilized.

These message packet delivery systems were based on a standard called X.25. (Although they still exist today, most X.25 packet switching systems have been supplanted by a newer technology.) Thus, if you needed digital messages (not voice) delivered from one place to another, you could turn to X.25 packet switching services for your communication needs. Aside from supporting wired connectivity, X.25 turned out to be a reasonable technology for use with satellite communications as well. Most X.25 packet switching communication suppliers could provide a low-cost ground station, anywhere in the United States (or world), linked to their network (fig. 3–3). The X.25 protocol is serial, relatively low speed, asynchronous, and simple enough to even be implemented in the front-end processors of many SCADA host computers. At the RTU level, the network is essentially transparent. The PAD does all the hard work and handles the actual X.25 protocol. Several pipeline companies, with very remote field sites (i.e., pump stations in Alaska or Saudi Arabia), found this satellite connectivity ideal for their situation. X.25 networks must be specifically configured (and PADs installed), and they were not designed to transport IP message traffic, so they would not offer a viable cyberattack pathway. But X.25 packet switching networks are now considered as obsolete, legacy technology.

In the 1980s, the domestic telephone companies had finished transitioning from the analog world to the digital world. This meant that instead of sending an actual voice signal from one point to another, they *digitized* (using an A2D converter circuit) the voice as close to the source as possible (converting it into a stream of eight-bit binary numbers at the rate of 8,000 of these numbers per second [if you do the math, this adds up to 64 kilobits per second]). These binary numbers could then be treated just like any other type of data—transmitted from one computer to another, stored, copied, and so forth. When they arrived at their destination, the numbers were reconverted into an analog signal via a *digital-to-analog* converter circuit. If you want to hear that process in action, just pick up your cell phone and make a call.

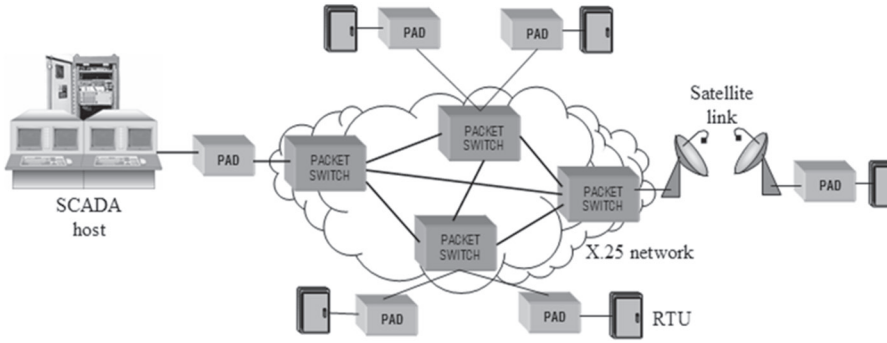


Fig. 3–3. Use of packet switching networks for SCADA communications

The digital telephone company

Today, the digital telephone on your office desk (or a card in the PBX system) performs the digitization (and reconversion to analog) of your voice, particularly if it is using *VoIP* technology; thereafter, your voice exists only as a stream of binary numbers, until reconstituted by the telephone of the person you are calling. (Home telephones still remain mostly analog, although with a digital subscriber line [DSL] circuit, the digitization happens in the DSL MODEM supplied by the telephone company.) Since the underlying systems of the telephone company are now just moving binary numbers around, why limit that to numbers that represent voice? Why not move numbers that represent data files, text documents, and spreadsheets—in other words, generic data? In fact, today you hear the telephone companies talking about voice and data services. That is because it is really all the same to them, just moving binary numbers from point A to point B.

It actually takes extra circuitry and fuss for the telephone company to create the effect of an analog circuit for companies that need them for their SCADA systems. The telephone company would much rather provide digital circuits, and they can offer several types. Remember that the telephone company deals with fixed data communications services (they also deal with mobile, but that mainly means cell phones). "Fixed" means that you need to know in advance all of the locations where you want communication connections, and you can't change these on the fly, once the connections are established. (You can always go back to the telephone company and march through the paperwork and process to have new circuits added and things changed, but it takes time.) Since SCADA host computers and RTUs tend to stay put once installed, this is usually not a problem for SCADA applications (fig. 3–4). Remember that the majority of telephone company data transport technology is predicated on setting up and maintaining semi-permanent, point-to-point (and multi-point) connections and not on providing dynamic connection networking. They create a private set of network connections to your designated locations, within their overall telecommunications infrastructure. One advantage

of this is that an external attacker has no easy way to connect into that same infrastructure and attempt to attack your systems, unlike with the Internet. Also, the service provided to you is a link-layer service so there are physical hardware addresses, but no IP addresses, involved. The only endpoints you can see/reach are the ones the telecom service provider has setup for you. Normally you place a router, with the necessary circuit interface card, at each location to connect to the phone system circuit. Depending on the service type and circuit type, the router will handle the link layer and hardware issues. This is also called "private line" service or multichannel multipoint distribution service (MMDS).

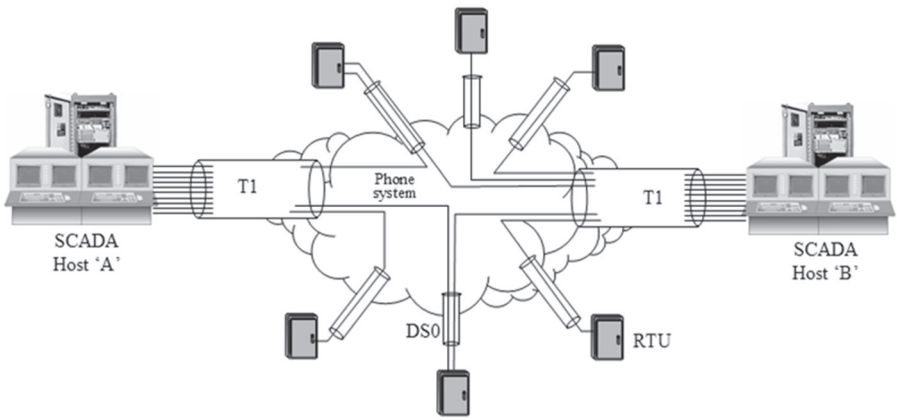


Fig. 3–4. Connection-oriented telephone circuits

T1/T3 circuits

The various regional telecommunication providers have been offering "private line" digital connectivity for several decades, particularly before broadband Internet connectivity became so prevalent. If you had a company (or SCADA system) with multiple operating locations, you could arrange for the telecom providers to bring a digital communication circuit to your various sites, where you would use a router (with the applicable interface module) to connect to that circuit and turn it into Ethernet that could be distributed to your various computers. The routers at each endpoint would be configured to send their traffic to each other based on the IP addresses of the messages. In the case of SCADA, you might want a T3 circuit (44.736 Mbps) for the host system whereas at each RTU location, a fractional-T1 circuit (64 Kbps to 1.544 Mbps) might suffice. Over time, various physical networking technologies have been offered for this purpose (aside from actual T1/T3 circuits), including ISDN, frame-relay, FDDI, and ATM. If at the primary operating location you also wanted Internet connectivity, the telecommunication provider (who was now also your ISP) could split off part of the T3 bandwidth and route it to the Internet (but you

really don't want your SCADA system connected to the Internet). You paid a fixed monthly lease fee for each such endpoint—the faster the connection, the higher the monthly cost. The telephone system is *not* providing any networking services, just point-to-point connections between routers. In effect, they were setting up a "private" network for you that was layered on top of their physical infrastructure. You can layer TCP/IP onto this communication (physical/data-link) layer as long as all of the computers and RTUs support IP networking. Figure 3–5 shows an architectural theme that will be repeated frequently when we discuss IP networking (and taking IP out to the field sites), regardless of the type of wide area network used to connect the various sites and whether you build it yourself or rent it from a telecom service provider.

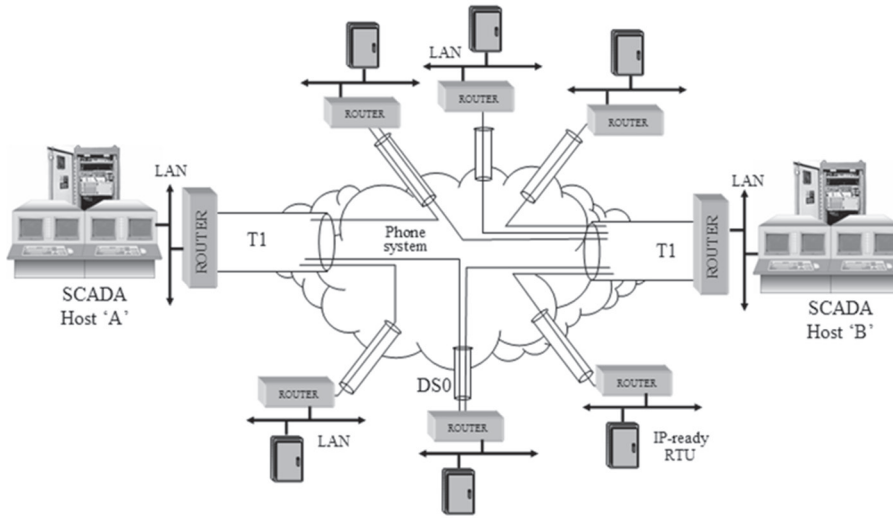


Fig. 3–5. Using digital telephone circuits to bridge LANs

Integrated service digital network (ISDN)

At the very end of the 1980s, when the telephone systems were still using mainly analog lines for home and small office services, and PCs were becoming a standard fixture in most offices, the telephone company introduced a higher-speed digital connectivity option that eliminated analog MODEMs and offered up to 128 Kbps data transmission rates. The integrated service digital network (ISDN) was the first digital offering by the telephone company. Today, this technology has been totally eclipsed by other telephone company offerings, at least in the U.S. It was never really adopted for widespread use in domestic SCADA systems (more so in Europe), but was frequently used for linking information technology (IT) systems with branch offices (providing a dedicated connection between the LANs at two sites). The phone company called it a *SOHO* (small office-home office) solution. It

was also more popular in Europe than in the U.S. The Internet was far less accessible than it is today, and so this was a technology for private networking through the telephone system. Today, a DSL line, which actually connects a customer onto the Internet, would be the technology of choice for equivalent purposes (as long as your firewall and anti-virus protections at each end are being kept up to date).

Frame relay

The technology that the telephone company preferred to offer to those who wanted data services back when this book was originally published (that is, modest bandwidth data services to many locations that don't involve Internet connectivity) was frame relay, which in architecture resembles an X.25 packet switching network but offered much higher performance at a slight reduction in reliability. The X.25 packet switching system was designed to link a large number of geographically dispersed dumb terminals with remote mainframe computers. These dumb terminals could not execute programs because they had no computer components. They were much like the early dumb RTUs; they could send short binary messages in response to users hitting keys on the keyboard. Thus, the X.25 PAD had to collect messages from a large number of these terminals, form them into an X.25 message, and send them over the network to the destination. The PAD retained a copy of this message until delivery had been confirmed or sent it again if delivery failed.

Today, we have no dumb devices. The RTUs (and modern point-of-sale terminals) are computer-based and able to perform many of the functions performed by a PAD. This means that the delivery security and routing features built into the X.25 networks could be discarded in favor of better performance. As telephone companies pushed SCADA system owners away from conventional analog leased lines, the technology they were offering as an alternative (in the 1990s) was usually frame relay. A frame-relay system is basically connection-oriented: you designate points where equipment will be connected into the frame network and the bandwidth you want for each connection, and then the frame network is configured to set up these *permanent virtual circuits* (PVCs). Frame-relay technology advanced to a point where a lot more flexibility was possible, but it is not a totally dynamic networking technology. Frame relay was intended as a computer-to-computer multipoint networking technology and has its own link-layer protocol and addressing scheme. It was the first replacement offering to eliminate an obsolete X.25 network.

For SCADA systems that needed to replace their analog leased telephone lines, frame relay didn't totally solve the problem because their RTUs could not support frame-relay protocol. However, there were vendors of devices called FRADs (frame-relay access devices), which provide similar functions as PADs did for X.25 networks. A FRAD is essentially a router with frame-relay protocol support and one or more serial ports that can be bound to corresponding serial

ports on other FRADs elsewhere on the frame-relay network. A SCADA system could use a group of FRADs interfaced to the SCADA system's front-end processors, to provide the equivalent of the telephone lines it used previously, and a FRAD at each RTU location, to complete the virtual circuits (fig. 3–6). The effect is as if actual dedicated serial circuits were run from the ports on the SCADA system to the RTUs (which is why this ability of frame relay is referred to as a Permanent Virtual Circuit). More advanced FRAD product offerings even support VPN technologies for communications security. The connection to the FRAD is generally a serial (RS-232/C) circuit; therefore, with older RTUs that have built-in analog MODEMs, this unfortunately presents an interfacing problem. We will return to frame-relay technology later in this chapter, but it should be noted that a few years after this book was originally published, telecom providers started dropping support for frame-relay services. We are discussing it because some older SCADA systems still use legacy frame-relay communications.

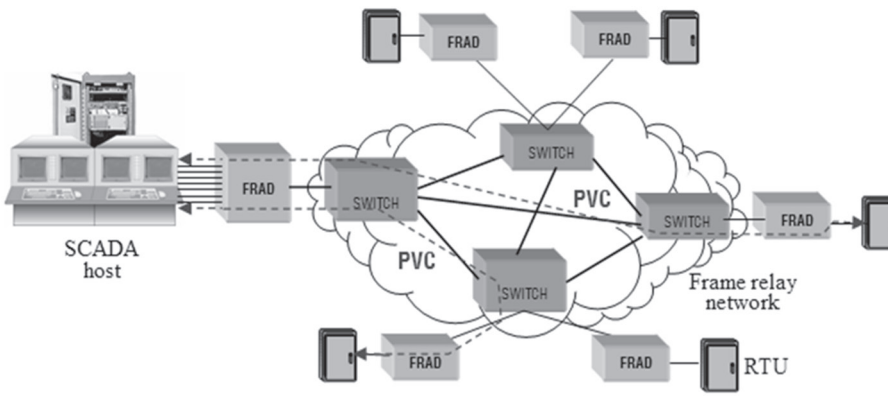


Fig. 3–6. Frame relay and FRADs used to replace analog leased lines

DSL technologies

Your local telephone company can probably offer you some form of DSL circuit, depending on how far you are from the nearest exchange or central office, but this is not a general purpose point-to-point (or multipoint) circuit. This is a dedicated circuit that offers you a voice-grade analog phone line and reasonably high-speed connectivity (typically *unbalanced*) directly onto the Internet. There have been successful experiments in California involving moving real-time data from a large population of geographically dispersed field sites, to SCADA host computers, over the Internet. In this instance, each remote site used a DSL circuit for its Internet connection, and VPN software, running in all of the RTUs and the SCADA masters, was used to provide communication security. However, many

SCADA system owners would probably shy away from considering the Internet as a secure telecommunications technology, particularly with all of the publicity concerning viruses, hackers, and attacks on computer systems. This doesn't mean that a SCADA system can't use Internet technology; private corporate WANs today are all IP-based, and those same networks can be extended to the field. As a general note, the Internet communications infrastructure is actually operated by the telecom companies and the same switches, wires, and fiber-optic cables that are delivering voice and data are also carrying all of the Internet traffic as well. There is little, if any, separate Internet infrastructure.

Options for Wireless Communications

Conventional licensed VHF analog radio (and its lack of security) has already been discussed. An alternative to that technology is unlicensed, spread-spectrum radio. Spread-spectrum radio gets its name because it transmits data on a range of frequencies (channels), rather than on a single frequency. Unlike licensed radio, where the user typically gets exclusive use of a specific frequency in their geographic area, all spread-spectrum radio equipment shares a common range of frequencies (bands with multiple channels) in one of three UHF frequency ranges, regardless of proximity to other spread-spectrum radio systems. This is possible because of the use of either frequency spreading (*direct sequence spread spectrum*) or *frequency hopping,* a technique that causes the transmitted information to be spread out over multiple or even a large number of channels. With frequency hopping, the radios jump from channel to channel (the band is divided into multiple channels), following a mutually agreed-upon, pseudo-random sequence (fig. 3–7). The purpose of this is to allow message-retry if you happen to try transmitting just as another nearby radio source does so as well. (There are two categories of hopping: fast and normal. In fast hopping, each bit is transmitted over multiple channels to make sure it is received.) There is a second type of spread-spectrum radio called direct-sequence spread spectrum, which adds synthetic noise (called a chipping code) to the transmitted data so that it seems like random noise, rather than an actual data-carrying signal, and this spreads the signal over multiple adjacent channels. Both types of spread spectrum technology make it more difficult (but not impossible) for an eavesdropper to listen to communications traffic but they should not be thought of a providing cyber security for the communications. The spread spectrum technologies are mainly used to improve communication reliability and decrease the impact of interference sources and really should not be thought of as security mechanisms. Wireless Ethernet (Wi-Fi) for example, uses direct-sequence spread spectrum, but still relies on additional security mechanisms (e.g., WPA2) to provide message confidentiality and integrity.

Although initially envisioned commercially as a local area wireless technology (it is used in cell phones, wireless Ethernet, and Bluetooth-enabled devices),

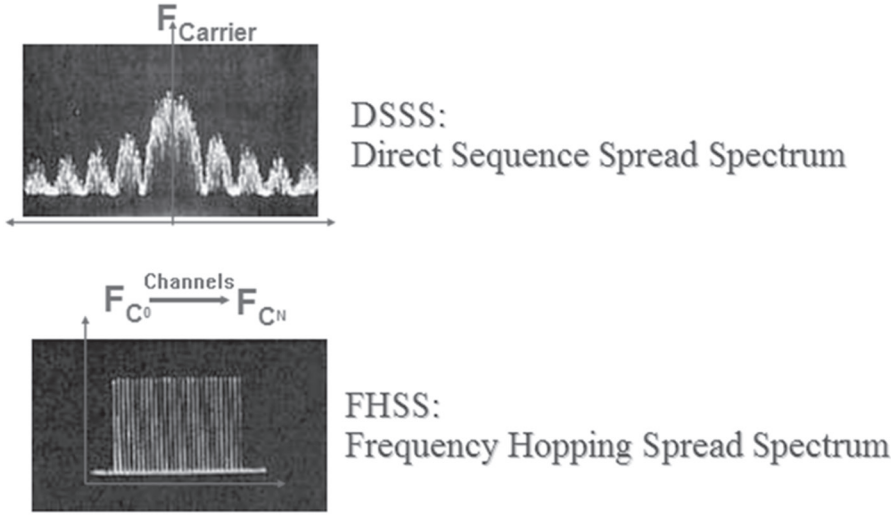


Fig. 3–7. Spectral energy (frequency) distribution of spread-spectrum radio

some spread-spectrum radio equipment on the market today is higher-powered, with transmission ranges comparable to conventional radio (10–20 miles, line of sight). Also, since unlicensed spread-spectrum radio is relegated to three ultra-high frequency (UHF) bands (900 megahertz plus 2.4 and 5.8 gigahertz), it can support much higher data rates (144 Kbps to multiple Mbps, typically) than does conventional HF or VHF radio. These frequencies fall into the international ISM (industrial, scientific and medical) band allocated for unlicensed general use. Spread-spectrum radio has been used by the military for years, owing to its much greater reliability; it is much harder to intercept and to jam than conventional radio. The problem with these higher frequencies is that they tend to require line-of-sight between stations. In a municipal area, that may mean having to place repeaters in many locations with good visibility (e.g., on tall buildings, water tanks, radio towers, etc.).

Almost all new SCADA system installations based on radio communications utilize spread-spectrum radio (because of no licensing requirements), unless the owners already have an investment in (i.e., an installed base of) conventional radio equipment. However, just using spread spectrum radio is not, in and of itself, sufficient for communications security. It is possible to intercept spread-spectrum messages. New stations can join a frequency-hopping radio network because every pseudo-random hopping pattern always jumps through a base channel as part of the pattern, so new stations can synchronize and join. Thereafter, they hop along with every other station and thus can communicate with them (or just listen to message traffic). If you know the manufacturer and model of a frequency-hopping radio, you can determine its available channel-hopping patterns. Fortunately,

many of the wireless technologies are designed to transport IP messages and thus can utilize IPsec and VPN technologies.

WiMAX

One particular category of spread-spectrum radio that has been standardized and commercialized is IEEE 802.16, or WiMAX, which is now commercially available (and is in fact the basis for 5G cellular service). WiMAX is essentially high-power wireless Ethernet capable of use over a large geographic area (e.g., for creating a municipal area network [MAN]) and is intended to provide high-bandwidth data rates (up to 244 Mbps) over municipal areas. The standard is designed for both mobile and fixed point-to multi-point service and consists of several subsections for different applications. WiMAX is Ethernet-ish and thus is intended for computers that have digital networking capabilities. It includes advanced encryption and authentication capabilities and supports point-to-multipoint communications. A modern IP-ready SCADA system, with IP-ready RTUs, would be able to use this type of communications infrastructure for real-time polling and supervisory control and still have lots of bandwidth left for the deployment of other applications and systems, such as voice over IP (VoIP) and security monitoring. WiMAX technology has been used, for example, in offshore production fields to provide network connectivity over the entire field.

Currently, building and maintaining a WiMAX system would be the responsibility of the SCADA system owner, just like any other radio communications system. However, in the future, municipalities may construct WiMAX systems to offer Internet connectivity to inhabitants and visitors and for their own municipal networking needs. SCADA system owners operating within the WiMAX service area will probably be able to utilize these systems for their communications (although security and reliability may dictate having a backup strategy). The downside of this technology is that anyone in the service area who has appropriate commercial hardware and software will also be able to use this wireless MAN and could attempt to attack the SCADA host or RTUs unless the available cybersecurity protections (such as VPN) are properly applied.

Cellular telephony

The local telephone company offers wired communications options, as discussed earlier. However, they also offer wireless communications options. The various digital cellular telephone systems installed in major municipal areas of the United States can be used for data communications, as well as for voice communications. As mentioned previously, a modern cell phone actually sends and receives data that represent digitized speech. The cellular system is a wireless mechanism for putting data onto the wired (digital) telephone system, where it can be transported in a conventional manner (possibly to another service area where it might

leave the wired telephone system and get onto another cellular system). Cellular providers can also provide short message service (SMS—messages up to 140 bytes in length), which can be applicable to the kinds of poll-response messages used in a typical SCADA application.

For SCADA systems installed in a municipal area, such as water/wastewater, electric power distribution, or gas distribution, the cellular telephone system offers a flexible means for establishing field communications. RTUs can be equipped with commercially available cellular MODEMs that can be dialed just like any conventional telephone number by a SCADA host. It is also possible to have the equivalent of leased connectivity (non-dialed, continuously connected) via the cellular system (fig 3–8), in which case a wireless connection is maintained continuously; the cost for this type of service is based on traffic (messages sent/received), and the telephone company routes this traffic back to the SCADA system via conventional wired (or wireless) means.

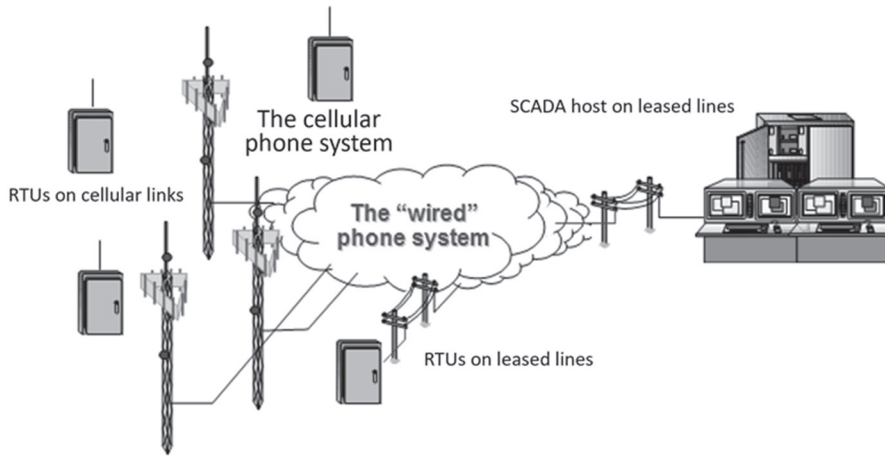


Fig. 3–8. Cellular data communications architecture

Most cellular service providers charge by the amount of data transmitted and received; thus, this technology best suits RTUs that employ a report-by-exception protocol design. Cellular gateways, routers, and MODEMs come in two variations: one that uses the cellular phone system to make point-to-point (station-to-station) connections—just like making a cell phone call—and the other that actually routes the traffic onto and back from the Internet—like using the Web browser on your cell phone. Depending on their design, they can either use a USB port to connect to the SCADA system or RTUs or an Ethernet interface (although some routers also offer a serial, RS-232 interface as well). The devices offer a wide range of communication speeds, and most are designed for low-power applications such as a solar-powered RTU application. One of the

challenges with using cellular communications is that there are several incompatible cellular technologies used by the competing cellular service providers. A CDMA-enabled cellular MODEM (or cell phone) won't work on a system that uses GSM technology, and neither will it work on a GPRS-based network (three of the incompatible cellular technologies used today). The good news is that most of the current cellular MODEM, gateway, and router products are capable of running on any of the systems used in North America, but separate equipment would be needed elsewhere in the world (where everyone but we and the Japanese uses GSM). Regardless of the technology you use, the folks that invented digital cellular recognized the need for confidentiality in voice communications and thus imbedded a reasonable level of link-layer stream encryption in the basic design of all cellular technologies. Of course, by using cellular technologies, you are relying on the cellular service provider(s) to maintain communication reliability and availability. Cellular connectivity may not be best as the primary communications mechanism for a SCADA system, but it is worth considering as a backup mechanism.

Digital Networking Technologies

Over the past few years, most SCADA system vendors have expanded their system capabilities to allow for communications between the host and RTUs via digital networking, specifically IP-based networking. This is partially because of the migration, by conventional telecommunications service suppliers, onto digital communications in place of analog communications. It is also because of the proliferation of available digital networking technologies and the widespread adoption of IP networking and the Internet. One of the qualities of most digital networking technologies is the ability both to support multiple, concurrent communication sessions (conversations) over a single physical communication link and to provide a flat communication architecture. In other words, any computer can communicate directly with any other computer, if this is needed (as when smart RTUs exchange data directly with each other or with multiple hosts). These are both useful capabilities for SCADA systems and are not generally available with conventional analog technologies.

There are many digital networking technologies available to a SCADA system owner, depending on the desire to use commercial suppliers or to construct a proprietary network. By network-ready SCADA systems and RTUs, we normally mean those that support IP networking with one of the currently available IP SCADA protocols: IP-Modbus, IP-DNP3.0, ICCP, or UCA2.0 or IP-based industrial protocols: OPC-UA, EtherNet/IP (CIP), ProfiNet, etc.

Frame relay

We have already discussed frame relay as an analog line replacement technology available from the telephone company, and this was true in the late 1990s and early 2000s, when this book was initially published. The problem is that by 2010, many of the telecom providers began dropping frame relay as a service option. Today, some providers are still supporting pre-existing customers, but no longer offer this networking service to new customers. AT&T was one of the largest providers of frame-relay network services and it has since eliminated that service as an option. Frame relay provided a "private" networking capability, but without throughput and QoS (quality of service) guarantees, but organizations began to realize that they could, for less money, get high-bandwidth connections to the Internet from a range of providers (including those same telephone companies) and then use technologies such as Virtual Private Networking (VPN) to make them secure. Frame relay was a private networking option where a given set of endpoints were connected through the frame network and connected to a device called a DLCI, which had a unique physical address. You could run an IP network over frame relay using the frame-relay WAN as your lower two layers of the communications stack. In fact, the ARP protocol could be used on a frame-relay network to get the DLCI address, given an IP address (or the reverse). Figure 3–9 shows a simplified setup for using IP-based communications over a frame-relay network.

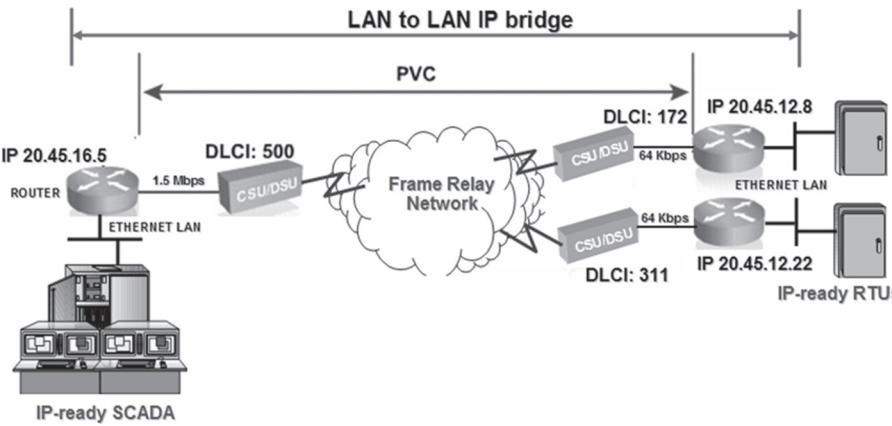


Fig. 3–9. Frame-relay DLCI-to-IP-address mapping in routers.

Today, frame relay is not readily available as a digital data service, as most telecom service providers have discontinued this service option in favor of providing DSL circuits (which, unlike frame relay, are not "private" circuits, as they route directly onto the Internet).

Fiber-distributed data interface

Another networking technology that is available to SCADA system owners is the fiber-distributed data interface (FDDI). This technology uses a counter-rotating fiber-optic ring, running at 100 Mbps, to move data from one point on the ring to other points on the ring. This networking technology was popular with small- to moderate-sized municipalities, universities, and large industrial campus facilities that wanted to create a fault-tolerant private voice/data network to link together all of their offices and facilities. If the municipality operated its own gas, electric, or water utilities, then this same network could be used to provide network connections for linking the SCADA host to its RTUs or PLCs. Because of the long overall distances that it can support (the dual ring can be up to 62 miles with single-mode fiber cable), the huge number of stations it can support (several hundred), and self-healing capabilities, FDDI has turned out to be useful for SCADA applications within 'wind-farm' (wind-turbine) electric power-generation installations.

FDDI, like most other networking technologies, moves messages from one local network to another local network. FDDI easily interfaces to Ethernet LANs and transports IP messages between these LANs. Because of its counter-rotating ring design, FDDI has the useful ability to self-heal if the ring is severed (which is always a distinct possibility with buried cables). Messages arriving in a repeater station on one ring could be sent back out the other ring if the repeater detects a link failure downstream (fig. 3–10). One of the issues with a shared network is the possibility that noncritical traffic will slow down the important messages (like when you are in a hurry and all those other people are clogging up the highway and slowing you down). FDDI networks are reasonably *deterministic,* which means that the time to deliver a message through such a network is always within a predetermined time span. The same can't be said for the getting onto and off of the FDDI network, through the switched Ethernet LANs (although by using the QoS settings properly, you can get faster throughput). But a FDDI network can support a significant amount of traffic, usually far more than enough for most SCADA applications. FDDI would be a choice if you elected to construct your own communication infrastructure.

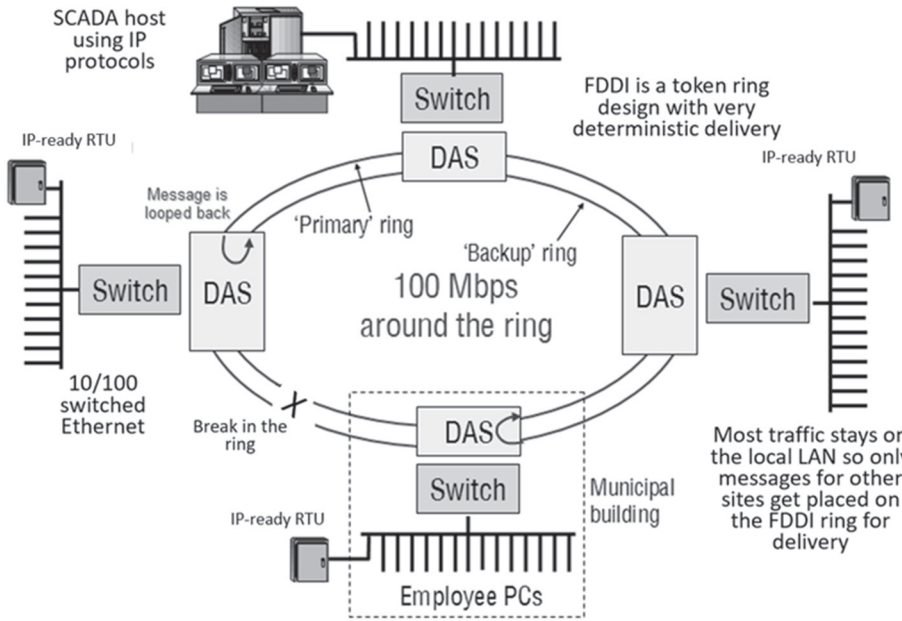


Fig. 3–10. FDDI counter-rotating ring design

Asynchronous transfer mode

Actually, the use of FDDI networking was just catching on, in the early 1990s, when another networking technology came onto the scene. Asynchronous transfer mode (ATM) equipment was invented by the telephone companies as their backbone technology for building and expanding the telephone system. ATM networks operate at very high data rates enabled by the underlying fiber-optic–based communications technology. Data rates of many gigabits per second (Gbps) are possible, and an ATM network can have a very flexible topology, including a self-healing ring like FDDI. The goal of ATM was to provide deterministic data-streaming services with guaranteed Mbps throughput that could support streaming video and audio as well as data streaming. An ATM network is set up with pre-configured virtual paths containing virtual circuits with specified bandwidth guarantees. Pre-configuration allows ATM networks to have minimal transport delays and highly simplified routing mechanisms.

Originally, ATM equipment was quite costly, particularly the top-quality (single-mode) fiber-optic cable used to make the physical communication connections between locations. But prices fell as fiber-optic cable became less expensive, particularly after the "dot-com" bubble burst. In addition, the bandwidth capacity of ATM goes well beyond that required for a SCADA system, so organizations that installed systems using this technology were able to underwrite costs by selling

or leasing their excess bandwidth. ATM networks can be small (e.g., a campus or small municipality) or huge, because of their scalable architecture. Although ATM equipment is still available, by the end of the decade of its introduction (early 2000s), this networking technology was already considered to be heading towards obsolescence. In fact, the ATM Forum, which had been promoting this technology, was disbanded in 2006.

In the same way that ATM technology eclipsed the FDDI networking technology about the time when it started to gain market traction, SONET has eclipsed ATM, particularly for large-scale applications. (Actually, SONET can be used to transport ATM cells [message frames] and also FDDI and Ethernet frames.) SONET technology was a further development by the telephone companies. SONET is prevalent in the USA, but a related technology called SDH (synchronous digital hierarchy) is used in Europe. The underlying networks that form the basis for the Internet are ATM and SONET/SDH networks, as are the networks used by the telephone companies to transport voice. (In fact, they are usually the very same networks.) Some electric utilities and pipeline operating companies have adopted fiber-optic ATM or SONET technologies for building or upgrading their long-haul communications infrastructures. Laying a fiber-optic cable in the trench with the pipeline, or stringing it along the transmission line conductors, has become a common practice.

TCP/IP Networking

The IP suite of protocols (including TCP) was developed as part of the U.S. government's desire to link together the computer systems and facilities of various government agencies, the military, research laboratories, and universities. The original project that evolved into the Internet we know today was run by the Defense Department's Advanced Research Projects Agency (DARPA) and remained a private, government-owned network for many years. In the late 1980s, the Internet was *privatized* by the government, and the rest is history. Because the Internet was originally private and run by the Defense Department, the design omitted a critical factor: there were no security or confidentiality features included in the design. This was corrected in the past few years with the introduction of IP version 6 (IPv6). As of 2012, every computer or device actually connected to the Internet had to support IPv6 and now has a 128 bit unique IP address. (If you had an assigned IPv4 address, there is an algorithm to convert it into your corresponding IPv6 address, and vice versa.) But most of the private networks, such as corporate WANs and ISP networks, will continue to use IPv4 for the foreseeable future.

Just about every country in the world is linked to the Internet and provides Internet connectivity to their citizens. The Internet is unique in that it employs connection-less communications. This means that there is no need, in advance, to establish communication paths between computers that need to exchange messages. In fact, such predefined paths would be an impossible burden given the size

of the Internet today. Like using the dial-up, world-wide phone system, if you know someone's IP address, you can establish a connection with them whenever needed, over the Internet.

The Internet uses an addressing scheme, rather than specific connections, and all properly addressed messages tossed onto the Internet will generally be delivered to the intended recipient (but that's not actually guaranteed). In this way, the Internet is somewhat like the U.S. Postal Service or the telephone system. You don't need to know the physical location of, or means for getting to, a destination address. The post office takes care of all of that for you, as long as you provide a valid address (or a valid phone number). (It is often useful to use the postal service as a model for explaining how IP networking works.)

The technologies developed for and used on the Internet are not restricted to the Internet. Anyone can buy the components and software to build an IP-based network. In most corporations, the company-wide network will be an IP-based network (probably using IPv4). When you buy a PC or laptop computer today, regardless of the operating system you prefer, that computer will invariably come preconfigured with support for IP networking (Windows, Linux, and OS-X all have supported both IPv4 and IPv6 communications since about 2011.) All of the vendor proprietary networks developed over the years—like Digital Equipment Corporation's DECnet and IBM's Systems Network Architecture (SNA), among others—have been almost completely eliminated by IP technology. (Which doesn't mean you won't still find them being used in older industrial automation and SCADA systems.)

IP also stands for inter-networking protocol, with its purpose being to allow separate, autonomous local networks (that may or may not be IP-based) to be interconnected. Within an IP WAN, there may be numerous LANs that employ different LAN technologies. But, just as IP has driven out almost all other WAN contenders, Ethernet has just about eliminated all other LAN technologies. In most private IP networks today, you will find Ethernet LANs with clusters of computers, peripherals, and PCs, and these LANs will be interconnected using routers and long-distance communication technologies (e.g., frame relay or leased digital telephone circuits), transporting messages (datagrams) using the IP protocol.

In many cases, the long-distance communications technology will be the Internet itself (fig. 3–11). Any company that wants to transact business today is essentially forced to connect to the Internet, if only to host a Web site and provide email support. If critical control systems (e.g., SCADA systems) are connected via IP networking to the business and IT systems that have Internet connectivity, then these critical control systems are theoretically accessible over the Internet. That is the strength and the weakness of IP-based networking. Remember that any systems (that use IP networking) connected to any systems connected to any systems that are connected to the Internet are also connected to the Internet (poor grammar, but you get the idea). This is why cyber protective technologies and strategies are necessary to keep SCADA systems safe.

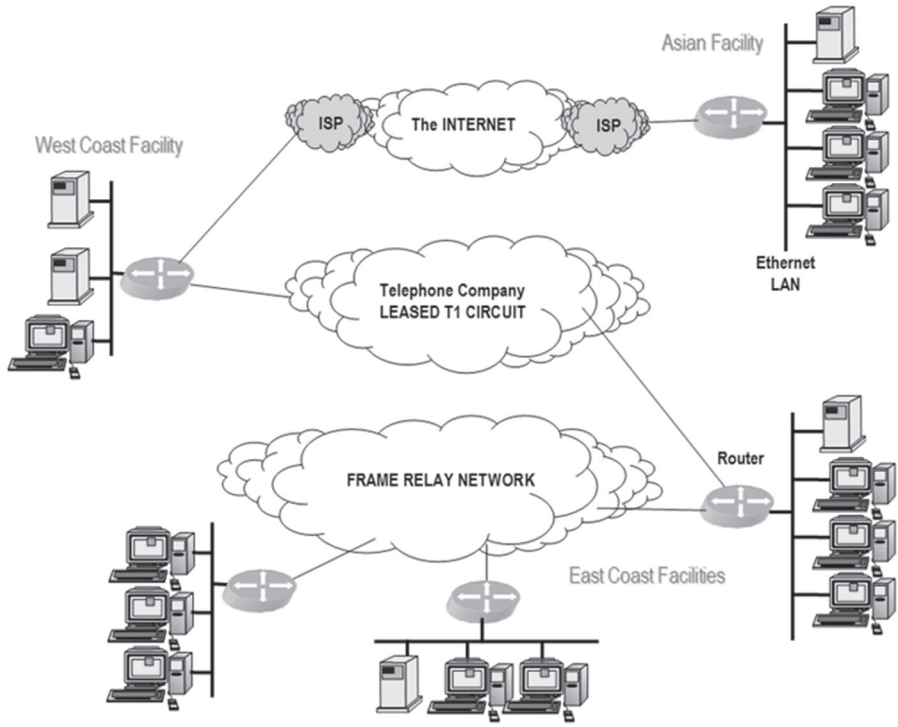


Fig. 3–11. Typical corporate IP network architecture

Every modern PC comes with Ethernet hardware and TCP/IP networking software. You merely plug your computer into a network switch port with a CAT5 cable and launch your Web browser to be on the Internet (admittedly, there are a couple of other steps, but it is no big deal requiring computer expertise). The good and bad thing about all of this (from a cybersecurity standpoint) is that the hardware and the software for IP networking (and Ethernet LANs) are readily available in commercial off-the-shelf (COTS) hardware and software. The means for connecting to and utilizing IP networks are available to anyone. Unfortunately, the Internet itself is home to innumerable Web sites that offer software and tips on hacking, creating viruses and worms, and launching attacks on computer systems. All of these *hacker tools* work just as well on a private IP network as they do over the Internet. (In fact, your IT folks probably use many of the same tools, to test your cyber defenses, that hackers use to attack them. We will be discussing such tools in a later chapter.)

Because SCADA systems are beginning to use IP networks for RTU communications and because these systems are often connected with, or are placed onto, corporate IP networks and exchange data with other systems (using IP networking and possibly doing so over commercial, public networks), it is important to

understand some basic concepts and terminology associated with IP networking. In a later chapter, we will discuss various vulnerabilities associated with IP networking and typical types of attacks used to penetrate and compromise computer systems. To begin, let us just establish basic terminology.

The fundamental communications protocol used to move all messages through the Internet (or any private IP network) is called, not surprisingly, the Internet Protocol (IP). IP delivers messages called datagrams from one host (computer) to another host, allowing intermediate computers (routers and switches) to assist in moving the message along to the final destination. In the case of LANs, IP moves messages between the various computers and devices but uses the physical LAN (typically Ethernet today) to transport the datagrams. Actually it is necessary to deliver messages between communicating *programs* running on different computers and not just to the computers themselves. A computer has an IP address (possibly more than one) and an Ethernet MAC address (possibly more than one), but how are programs specifically identified for message delivery?

For the design of IP to account for specific programs (also called *processes*) and not just computers, two transport protocols were layered onto IP: TCP (Transport Control Protocol) and UDP (User Datagram Protocol). Application programs can request either TCP interconnectivity/delivery or UDP delivery. TCP provides a more reliable connection, hides packetizing activities, and deals with all sorts of reliable-delivery issues. Two programs create a TCP "session," which exists until both programs terminate the connection (very much like making a phone call). UDP is a bare-bones delivery scheme that puts the burden of guaranteed delivery on the two programs that are communicating. In truth, UDP delivery is just IP delivery, with the addition of something called a port number. Unlike TCP delivery, UDP delivery adds no reliability mechanisms to the delivery process. UDP delivery is like sending a text message or a FAX; there is no persistence or assumption of ongoing message exchanges. The need for guaranteed delivery (and possibly an ongoing message exchange) or the need for rapid delivery (without a guarantee) are factors that could push the choice of TCP or UDP delivery one way or the other. In most applications, we want to be sure that messages are delivered completely and without errors and that a best effort is made to complete the entire delivery. TCP is the correct choice for transport service in that case. For this reason, it has become commonplace to think of the two protocols, TCP and IP, as consolidated, and thus the prevalence of calling it TCP/IP. But for some applications, such as streaming video or audio, it makes no sense to halt the stream in an attempt to resend a bit of data that was corrupted. In those cases, UDP is the correct choice for transport services. In an industrial application where, for example, a given data value was being resent several times a second to update a display, UDP delivery might be applicable and preferable, since any value received damaged would be replaced, a fraction of a second later, with a fresh value. TCP and UDP messages are carried as data (the *message payload*) in IP datagrams across the Internet or

local network. (If it is across a LAN, then the IP datagrams themselves are carried as data in the link layer protocol of the LAN [such as an Ethernet frame].)

One feature that these two transport protocols add is the idea of *ports*. Computers on an IP network have unique (IP) addresses (and in fact that can have more than just one), but there can be many different programs concurrently running on a computer and attempting communication with other programs running on other computers. Port numbers (a 16 bit integer value) were created to uniquely identify each communicating program. (Just like a street address for a building can include a specific apartment number so that the letter gets to the right person at the address, an IP address gets more specific with the addition of a port number, so that messages get to the right program.) There are over 65,000 possible port numbers as they are represented by a 16 bit integer (0 to 2^{16}-1). Unlike IP addresses, we have not run out of those numbers. For common applications (services) that are found on most computers, a standardized set of so-called *well-known port numbers* have been assigned (port numbers between 1 to 1023). Returning to the postal service metaphor for the moment, a port number is like specifying the specific person at an address to whom the letter should be delivered. A *well-known port number* is equivalent to using an addressee such as "Accounting Department" or "Human Resources" in place of a specific person's name. Since most companies have such departments, you can get your mail delivered, even if you don't know a specific name of a person in that department. Using a well-known port number is how you get connected to a computer's email server, Web server and most other common services. Unfortunately, because they are well-known, hackers can attack these ports and attempt to exploit identified vulnerabilities in these system-level services.

IP addresses (until IPv6) were just a 32 bit binary number. (That number is usually broken into four octets with each octet written as a decimal number separated by periods: e.g., 128.156.12.33.) The number actually has two parts: a network ID and then a unique computer/host ID within that network. You can think of this like having an area code and phone number. Other people may have the same phone number as you, as long as they have a different area code. The people who initially cooked up the Internet couldn't envision there ever being more than 2^{32}—1 computers connected on a single network worldwide. They were so very wrong. We ran out of unique IP addresses a few years ago, and there have been several temporary workarounds employed to keep things going. In IPv6, the address is a 128 bit binary number (and that should last us a while). When an IP message (actually called a *datagram*) arrives at a computer, the message headers contain the IP address and port number of both the destination computer/program and the sender computer/program. (Actually, it may arrive as multiple pieces called *fragments* if the original datagram was large, and each fragment will have that same delivery information.) This information is like the return address on a letter, the original purpose of which was so that you could send a return message using that address/port information. Today, that sender information might also be used to decide if the message should be accepted or discarded (that is called *packet filtering,*

and it is a low-level firewall function that is easily defeated). Unfortunately, it is all too easy to falsify (spoof) that information in an IP message. We will discuss the use of firewalls in a later chapter.

For most people, the mysteries of IP addresses and ports remain hidden, and the only Internet 'address' they are aware of is the uniform resource locator (URL) for the Web sites they visit and people to whom they send email. A URL is the string that you type into a Web browser or email system to specify a person or Web site on the Internet (or within a private IP network). Most people won't recall an IP address and port number but can remember something like *tim.shaw@ourcompany.org*. That text string is slightly more human-friendly than a group of numbers, but as previously mentioned, sending something over an IP network requires an actual IP address. For that reason, there is a special message protocol that can be used to look up an IP address when given a URL. On the Internet (and in private IP networks), there are computers whose function is like telephone company directory/yellow-pages services. They offer *DNS services*.

If you send a message containing a URL to a *domain name server* (using DNS protocol), it will look up the domain name (the "ourcompany.com" part of the text string) and send back to you the IP address for that email server. Most computers retain a local copy of this information (called a DNS *cache*) so that you don't have to ask for it again (like writing down the telephone number you get from the directory assistance folks). Next time you use that URL, the computer will just refer to its local copy. One way hackers attack computers is by overwriting the DNS cache so that your computer gets the wrong IP address. (Imagine someone changing the telephone numbers in your phone book, so you call them [the bad guys] when you think you are calling your bank or stock broker.) When we speak of IP, we are really discussing a huge and growing suite of protocols layered onto IP or existing down within the layers of IP itself (fig. 3–12). Many of these protocols are essential for making networks function, but most were also not designed with security features, at least in IPv4.

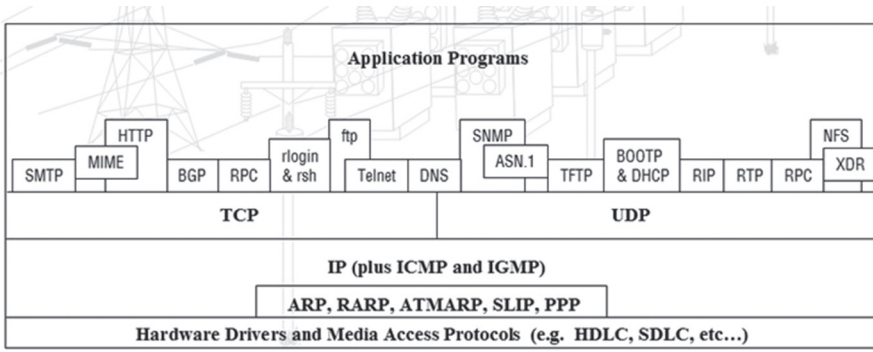


Fig. 3–12. Some of the basic protocols in the IP suite

Ethernet LANs

IP was initially intended as a protocol and mechanism for moving messages between local, autonomous (proprietary) networks. As mentioned earlier, Ethernet won the LAN war. Today, the vast majority of LANs are Ethernet-based, and IP messaging can be used among the computers on an Ethernet LAN, as well as between such LANs across a WAN. Every computer on an Ethernet LAN needs to have a special, unique address for that LAN (called its *media access control* [MAC] address). This is because Ethernet was developed independently of IP and has its own separate standards. To be connected to a basic Ethernet LAN, a computer is equipped with a circuit card called a NIC [Network Interface Card/Controller]. That NIC has a built-in 48 bit number (the MAC address) assigned by the card manufacturer at the factory. As with IP addresses, MAC addresses were supposed to be unique (but we ran out of those also), at least on a given LAN segment. When a computer wants to send a message, that message is put into a suitable format (an Ethernet frame) in accordance with the IEEE 802.3 standard, and that frame includes the MAC address of the sending computer and that of the destination computer. NIC cards inspect messages (the destination MAC address in the message header) to see if they are the intended recipient (there are also special addresses used for *broadcast* and *multicast* messages). If computers on an Ethernet LAN want to use IP networking, the IP datagrams (or datagram fragments) are carried over the LAN as data inside the Ethernet frames in the data area of the frames. In order for computers on an Ethernet LAN to send IP messages to each other, there needs to be a way to convert a MAC address to an IP address (and vice-versa). Fortunately, there is a protocol called the *Address Resolution Protocol* (ARP) that can be used to get an IP/MAC address pair by sending out special Ethernet broadcast messages. These address pairs are then kept in a local storage area (the ARP cache) inside your computer, so they can be reused when needed. As you might have already guessed, another way hackers attack systems is by overwriting the contents of the ARP cache and substituting a bad MAC address, so that your messages go to the wrong computer (just like the DNS cache example described earlier). This kind of attack (*ARP cache poisoning*) is used on a LAN for several purposes, a major one being to solve the hacking problem of being on a switched LAN that limits what message traffic you can see.

A continuous stream of new application protocols have been layered onto IP: instant messaging, voice over IP, video conferencing, on-line gaming, and Internet radio are just a few examples, and there are many more. Each of these new protocols and associated services typically claims one or more of the unassigned port numbers available with TCP and UDP (the service is registered and officially assigned a port number). This also creates potential security problems. Hackers can hide their attacks within these protocols or stage attacks by sending messages to a port that has not been properly protected. Some protocols (such as classic OPC, which will be discussed later) just pick unclaimed so-called *ephemeral ports* at random, which makes it hard to set up a packet filtering function in a firewall, since you don't know in advance what ports it will claim.

Along with new protocols, various security features have been added, to correct the lack thereof in the original IP design. In IPv6, full message encryption and end-to-end *authentication* (usually called IP_{SEC}) were added and also backfilled into IPv4, but enabling these capabilities (the ESP [encapsulated security protocol] and AH [authentication header] extensions) is optional in IPv4. Authentication and cybersecurity technology will be discussed in detail in subsequent chapters.

IP networking involves a multilayered network architecture, with a great number of specialized protocols that perform the background functions that make all of this stuff work. Just like a successful Broadway play requires a huge support staff working in the background, IP networking depends on a large number of background functions and layer-to-layer messages.

Secure Socket Layer

Because the Internet was initially designed with no inherent security/-confidentiality mechanisms, creative vendors had to devise ways to accessorize TCP/IP networking to make it suitable for major applications such as the use of Web server/browser technology (the basis for the World Wide Web) in electronic commerce. People would not be willing to conduct financial or other sensitive transactions with Web sites, using a browser, unless a means was provided to ensure the confidentiality and integrity of the transaction and of the financial information necessary to complete such a transaction.

Layering-on Securtity (SSL)

Netscape devised a means for adding customer-to-Web-site security by utilizing *digital-certificate*–based authentication and *session-key* message encryption. This mechanism was called the *Secure Socket Layer* (SSL). This was not part of the TCP/IP system design in IPv4, but rather something added on top of the formal TCP/IP mechanisms (which is why they called it a layer). A Web site and browser that supported SSL would perform a series of security set-up operations—after establishing a TCP session, but *prior* to the conventional *HTTP* Web page delivery—and then all further transactions between the client browser and Web site/server would be subjected to message encryption. SSL became a de facto standard because it worked, solved a major problem, and didn't upset the underlying TCP/IP networking mechanisms. The problem is that SSL functionality needed to be added to applications because it was not part of TCP/IP. Today, every Web server program and Web browser application comes with SSL functionality already installed. Some other client-server combinations (e.g., for email delivery) also come with SSL installed. Within the SSL process, message encryption is performed to provide confidentiality. Encryption requires the use of an *encryption key,* which is just a big binary number used to scramble and then descramble the message. SSL uses a special form of encryption key to setup the process: a key pair composed of a *public key* and a *private key*. It can also use something called a *digital certificate* to perform authentication of the Web site. These

topics will be covered later in this book. For now, just recognize that SSL is what makes it reasonably safe to enter a credit card number on a Web form.

VPN

Another mechanism that was devised to layer security onto IP networking is the virtual private network or VPN. This technology is similar in nature to SSL in that it incorporates a set of authentication and validation functions layered on top of IP and includes message encryption and uses *digital certificates*. The major difference is that SSL was specifically designed for secure Web-browsing activities (and built into those specific applications), whereas VPN technology can be implemented external to the computers whose traffic it is protecting; it creates a secure network pathway that protects all forms of IP application messaging. VPNs can be created external to your computer (e.g., by a pair of routers) so that it is transparent to communicating applications. VPNs also use message encryption algorithms, encryption keys and digital certificates. For most of us, our first exposure to VPNs is when we are traveling on business and need to connect to our business systems over the Wi-Fi/Internet connection at the hotel where we are staying (fig. 3–13), using a so-called remote-access VPN.

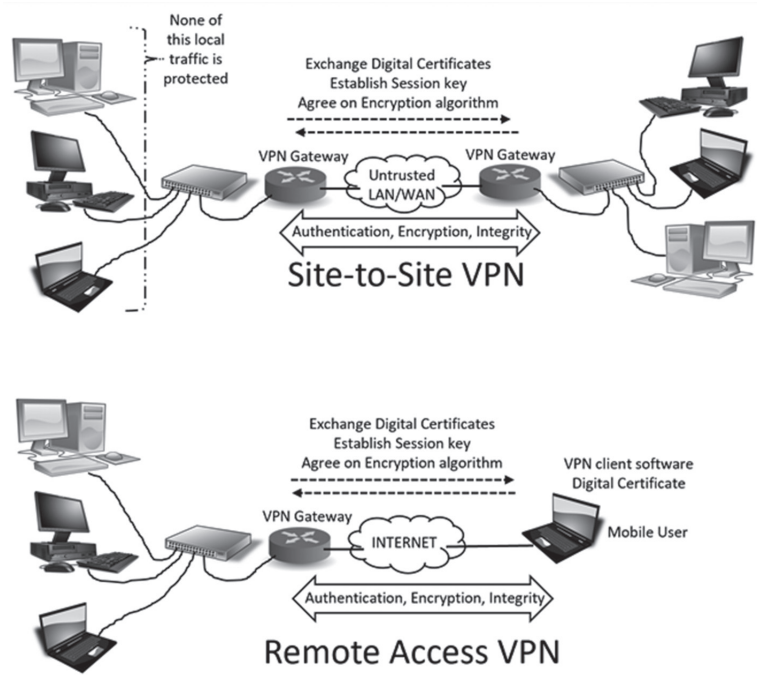


Fig. 3–13. Site-to-site and remote-access VPNs

There are three basic forms of VPN: the permanent, site-to-site VPN, the VPN within a LAN or WAN, and the temporary, remote-access VPN. Where there are two (or more) sites with numerous computers that need to exchange message traffic between and among the sites on a regular or continuous basis, a permanent site-to-site VPN (sometimes called a VPN tunnel) can be created by setting up VPN functions (authentication, encryption, digital certificate validation, etc.) in the routers that interconnect the sites. In this instance, the routers are performing a *VPN gateway* function. Communications among the computers on the local networks at the sites themselves are not protected by the VPN, but if any of the communications traffic flows outward through the local router to an external network (e.g., the Internet), that traffic is protected by cryptographic measures until it reaches the opposite router at the distant LAN. In effect, a safe tunnel through the external networks is created by the routers (which is pretty important if that external network is the Internet). Setting up a tunnel like this is transparent to the existing computers and applications; they don't need any modifications to take advantage of the tunnel. With VPN within a LAN or WAN, every computer that is a VPN member fully implements all of the necessary protective functions so that all traffic among the members is protected and connections authenticated. (For example, if every computer in the Accounting Department supports IPv6 and is issued digital certificates that define them as being VPN members (and for authentication and encryption purposes), then you have created a VPN within your corporate WAN.) This is more complicated than setting up a tunnel, and it may not be necessary as long as you can trust the security of your LANs. A common variation on these two approaches, where you have remote or mobile users that need occasional access to your systems, is to use a VPN gateway that connects to the Internet to allow remote PC users to make temporary connections to your LAN, through remote access (via the Internet). Each remote PC implements VPN client software, and there is essentially a private, individual tunnel between each remote user and the VPN gateway. But, once the traffic from any such PC passes through the VPN gateway and is placed onto the LAN, it is no longer protected. Remote-access VPNs are a common means for mobile workers to get access to corporate assets in a secure manner, while connecting to them across the Internet and other public, un-trusted networks. The VPN gateway also protects against attacks coming from the Internet since attackers (in theory) won't know the private key of the gateway and thus won't be able to authenticate and set up a TCP session.

It is common for a SCADA system to have one or more IP-based WAN connections with other organizations and/or to an alternate operating facilities. The connections between the (primary) SCADA facility and the other location(s) should always be protected by the use of site-to-site VPN technology. If the connection does not use IP-based networking then VPN isn't possible, but then the communications are probably proprietary and far less likely to be usable as an attack pathway. If the connection is IP-based and if the other organization will (can) not support VPN at their end, then the least the SCADA end should have for

its protection is a firewall. If the connection is over a private or leased WAN, then a well-implemented firewall may be adequate since access to the WAN would be difficult to achieve. If, however, the WAN is the actual Internet, then a mere firewall is probably not adequate, and a NIDS sensor should be used to monitor for malicious message traffic. In that case, it would be better to use a more advanced firewall (e.g., a UTM firewall) at the SCADA system end. One of the biggest challenges, even with a VPN connection, is the possibility of compromise of the computer system(s) at the other end of the WAN connection. With a connection from the primary SCADA facility to an alternate operating facility, the SCADA system owner has control over the physical security of the alternate site, as well as the cybersecurity of the backup SCADA system at that site. With a connection to another organization, even when VPN is used, there is no guarantee that an adversary might not be able to compromise the computers of that organization and then use them (via the VPN protected WAN connection) to attack the SCADA system. Therefore, in those instances, it is important to further protect the SCADA system by the application of both a firewall and a NIDS sensor (fig. 3–14). If the Internet is the WAN that connects a primary SCADA site to the alternative site, then both sites ought to employ VPN gateways and use a firewall and NIDS sensor to block and detect attacks coming from across the Internet. With an Internet connection, it is always preferable to use a more advanced firewall and not just a simple stateful packet filter device, even when a site-to-site VPN is being used.

The Internet

Although today it is still uncommon to see the actual Internet used as a communications system for real-time supervisory and control applications, there have been successful demonstrations of this capability. As SCADA host computers and RTUs with IP support become widely available, it is likely that the argument will be made, if purely for financial reasons, that the Internet should be considered. Thus, some knowledge of the Internet (as opposed to just IP networking) is essential. As with IP networking technology, the Internet is too complex a subject to cover in any detail within this book (or in 10 books, really). So, we will limit our discussion in this chapter to architectural, reliability, and performance considerations; security issues related to IP networking and the Internet will be discussed in a later chapter.

Internet backbone

The Internet itself is composed of a number of independent but interconnected computer networks linked by very-high-speed communication links. Within the United States, this network is operated by a number of the regional telecommunication and telephone carriers. These computer networks provide the routing

and message-forwarding mechanisms that carry IP messages (datagrams) from one autonomous computer network to another. As was mentioned previously, those telecommunication carriers have built SONET, ATM, and other forms of IP-compatible digital networks (to provide voice and data services), and they can be linked using gateways that pass the IP datagrams from network to network. The Internet is actually composed of this collection of interconnected, autonomous networks with all of them supporting IP networking (now at IPv6). An autonomous network could consist of one computer, but usually they are more like your own company network, which may have only one actual connection to the Internet (via your ISP). Most of us will never actually make a connection to the Internet; we will make a connection to the autonomous network operated by our ISP. As was mentioned earlier in the chapter, the ISP's network probably still uses IPv4 but has one or more connection points to the Internet backbone where the ISP provides *NAT* services (network address translation), which just means they convert your IPv4 messages and addresses into IPv6 messages and addresses, which can cross the Internet. There is a lot of communication bandwidth built into the backbone, as well as a lot of fault tolerance. (It was initially intended to survive a nuclear attack.) Direct interconnection to the backbone is restricted and is almost never available to a commercial user. Most direct backbone connections have been to governmental organizations and facilities, major research laboratories, ISPs, and major universities with governmental research programs (and to the Internet networks set up and run in other countries).

The initial design of the Internet did not address issues of real-time message delivery or of any kind of streaming message traffic. But today, people want to stream audio and video across the Internet and use it for point-to-multipoint video conferencing. IPv4 had no real ability to handle such streaming activities; fortunately, IPv6 has several mechanisms that support streaming, and that means we can now use them. But a basic fact of IP networking, even with IPv6, is that message delivery is not guaranteed. IP tries its best, but in the end the message (datagram) may not get delivered (although you might at least get a message telling you your message was tossed out). This is actually the reason why using TCP delivery is important.

Internet service providers

Surrounding the backbone are Internet service providers (ISPs) that do have direct, high-bandwidth backbone connections and that sell full-time, moderate- to high-bandwidth connectivity to others, such as yourself. Most of the original ISPs were the same regional carriers that run and maintain the backbone systems. But today, other Internet service providers have emerged, including cable TV companies, cellular telephone companies, and satellite communication companies. The type of connection you make to the Internet and the type of ISP through which you connect have a lot to do with your connection reliability and responsiveness.

Today, you can usually pay for guaranteed levels of bandwidth from your ISP. Your ISP creates a local autonomous network (which includes your computer) and handles the gateway and NAT functions of connecting to the actual Internet backbone and routing messages to and from your computer. In most cases, your ISP also assigns you a temporary, reusable IPv4 address for each of your computers by performing *DHCP* services in the router/gateway (e.g., cable modem) they install in your home or office.

IPv4 and IPv6

For many years, IPv4 successfully enabled the Internet to function and grow, but it had limitations, a major one being an inadequate supply of assignable addresses. But on private networks, IPv4 still provides adequate performance. IPv6 introduced a lot of improvements aimed at addressing the shortcomings of the prior version. As was already discussed, IPv6 introduced a much larger IP address space and added the optional facility for built-in communications security (IP_{SEC}), among other enhancements.

Possibly just as important was the addition of performance guarantees for special quality of service (QoS) requests. Streaming real-time data over the Internet had been a problem holding back developments like voice (telephony) over IP (VoIP), video conferencing, streaming audio (Internet radio stations), and other applications. IPv6 added mechanisms for designating categories of message traffic that needed priority handling (quality of service and real-time service), so that end-to-end message delivery times can be kept under a minimally acceptable threshold. This would be an important consideration for any supervisory real-time monitoring and control application deployed across the Internet. SCADA system users need to know that if they click on a display to send a control command to the field, that control command will be delivered to the remote, and a response or indication received, within a guaranteed, acceptable time limit. Networks that can make such a guarantee are said to be highly deterministic. Until IPv6, the Internet (and IP networks in general) could not promise deterministic performance. If loading got heavy, response and delivery time could and would degrade.

As will be discussed in the next section, some forms of supervisory control models (e.g., pipeline *leak detection* and electric power grid *state estimation*) require a full field data update periodically from all remotes, within a narrow time span, in order to function properly. Within an IPv6 environment, this could be guaranteed, across any connection points on the Internet. Note that in a private IP network, where traffic could be controlled and limited, acceptably reliable message delivery times can also be achieved, even with IPv4. However, on the Internet, you have no such ability to manage the traffic loading from moment to moment. This is one reason why IPv6 enhancements for QoS are important. Nevertheless, remember the warning made earlier in the chapter: although the basic Internet backbone is now running IPv6, much of the equipment surrounding the Internet

(in private corporate networks, ISPs, universities, etc.) is still running IPv4 and thus cannot take full advantage of the enhancements introduced by IPv6. Only in 2004 did Microsoft upgrade the TCP/IP networking of their Windows operating systems to support IPv6 and its corresponding enhancements. Most of the IACS systems and products on the market today still assume an IPv4 network using Ethernet for any LAN portions.

4

Supervisory control applications

The software that runs on a SCADA system (host), plus the data and configuration information it needs to perform its functions, are essentially the target of attack when we are discussing cybersecurity. This includes the software that supports the operator consoles, software that communicates to the field equipment, and the configuration data (databases) these utilize. Although it makes for great special effects at the movies, it is not actually possible to cause a computer to explode, spray a shower of electrical sparks, or burst into flames by sending it secret commands. It is, however, possible to shut down or disable a computer, or destroy valuable information stored on the computer, or create a *denial of service* attack, by sending the right commands. When considering what the objectives of a cyberattack on a SCADA system might be, particularly given the complexity and difficulty of implementing such an attack if adequate cyber protections are in place, a logical assumption would be that the attackers want to cause significant damage. Just shutting down the SCADA system for a while might (or might not) achieve that objective; they may need to take control of the SCADA system and use it to manipulate the process (pipeline, transmission system, pumping stations, etc.) in a dangerous manner, or they may need to trick the operators into doing (or not doing) something that triggers a hazardous event. But in all three cases, this involves manipulating system software and/or system data.

The software that comprises a SCADA system can be described in terms of layers. The first layer is the software that owns the computer and its resources (memory, bulk storage, peripherals, network support, etc.) and makes them available to application programs. That is the operating system layer. For systems built in the last decade or two, that is most likely to be a Microsoft Windows Server variation or possibly a Linux variation. Today, it is also possible that the computer and operating system are staged in the form of a *virtual machine* and not an actual physical separate computer. The next software layer involves the basic SCADA system functions—including RTU polling and communications, basic display generation, operator interface, alarming and reporting, trending, and other fundamental SCADA capabilities. The top layer consists of the advanced supervisory application programs that make use of the collected information to perform more advanced calculations and potentially even send supervisory control commands back down to the RTUs in the field. Figure 4–1 shows this simple layering model. There may also be business applications and operational applications associated with the SCADA system, but typically these

are not part of the SCADA system, and the association is in the form of requiring data from (and possibly providing some data to) the SCADA system. I mention this because one possible attack pathway into a SCADA system that is well cyber protected could be compromising an associated system that communicates with and is trusted by the SCADA system (like a business system) that isn't well protected, and using it as an attack platform to compromise the SCADA system.

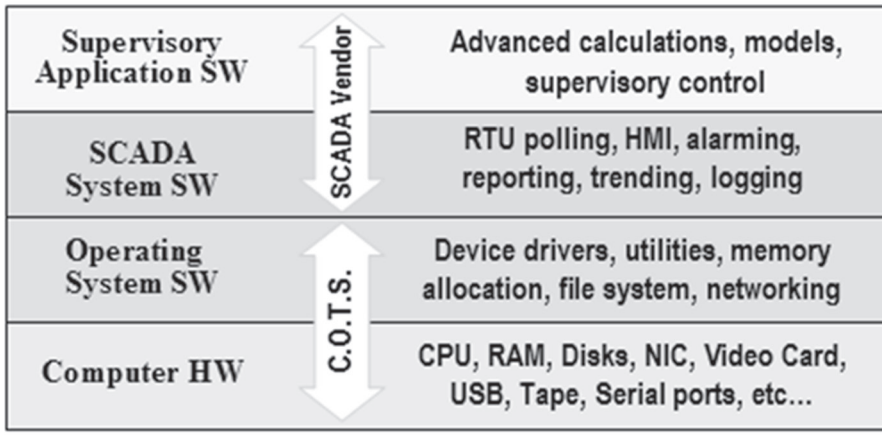


Fig. 4–1. Software layers comprising a typical SCADA system host

Operating System Utilities

In the early days of SCADA system development, there were many vendors of computer hardware, each with their own proprietary designs. Most did not supply an operating system, as we know it today, with their computer equipment. A SCADA manufacturer would receive a set of program development and debugging tools (text editor, assembler, compiler, and simple debugger) and possibly a simple set of utility programs (file utility, diagnostic routines, copy program, etc.). All of the software needed to perform SCADA functions and to control and manage the computer hardware was up to the SCADA supplier to develop from scratch. In the 1980s, this changed as computer manufacturers began to provide at least rudimentary operating systems—and then more advanced, multi-user operating systems—along with their hardware. From the 1970s up to the present, the overall amount of software and hardware supplied by a SCADA vendor has been decreasing (fig. 4–2). In fact, today most vendors mainly sell their SCADA software plus training and support services. Some may also offer a line of RTU equipment. But few still provide a turnkey product/solution and act as system integrators. Because of the migration toward standardized computer, networking, and operating system platforms, many more third-party applications are available. But this has also made the newer systems more vulnerable to cyberattack.

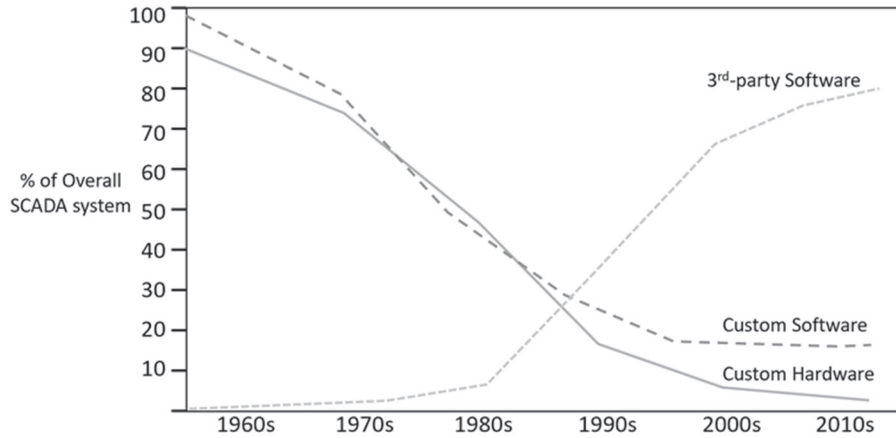


Fig. 4–2. Evolution of SCADA software with commercial software

Today, nearly every SCADA system sold runs on an Intel x86 or compatible computer platform, with either a Microsoft Windows or a Linux operating system. There are a few VAX/VMS- and Sun/Solaris-based systems still out there chugging along, but only until the spare parts are exhausted. This consolidation of technologies and platforms has a beneficial side: very good application portability; availability of third-party vendor, "best of breed" layered applications; readily available technological resources for software support; lower cost of ownership; and a good migration path for long-term system hardware support. All of these provide good, financially sound business reasons to explain why this consolidation has occurred. Unfortunately, many of these positives become negatives when considering the cyber vulnerability of SCADA systems. The problem is that there is a wealth of information publicly available about these operating systems and hardware platforms, including information about how to damage them, invade them, or cause them to malfunction. As SCADA systems more closely resemble our IT systems, the same methods, *exploits* and *malware* can be applied to attack them as are being used to attack our IT systems. (And keep in mind how often we hear about yet another business being hacked). Hackers trade tidbits and techniques on their Web sites or the *dark web* and sell 0day exploits to the highest bidder. So, you can assume that anyone desiring to attack your SCADA systems will have no difficulty obtaining expertise in its operating system, networking, and their various components, which form the underpinnings of the SCADA system. And you can probably also assume that they could afford to buy the same commercial SCADA system software that you are using, essentially creating a test platform on which to practice and explore new attacks.

Included within every commercial operating system are utility programs, system services and various application programs. (These should not be confused with the SCADA-specific system utilities, which are covered later in this section.)

Identifying exploitable vulnerabilities in these utilities, services, and applications is a major objective of hackers. This is particularly true for network-accessible services, as exploiting one of those can provide remote (cross-network) access into your systems. This is also why we will discuss removing or disabling any such services you really don't need as part of hardening your SCADA system. Installing new security patches is a full-time job these days. Microsoft dedicates a special Tuesday each month for releasing the latest patches. In the Unix/Linux world, intimate and detailed knowledge of your operating system is possible because the source code for all such distributions is publicly available. For the Microsoft world, there are tools to decompile the machine code distributed by Microsoft (as well as occasional pirated copies of source code, possibly illegally released by malicious insiders or international hacker groups).

A major reason that hackers would want to compromise a service running on your system is that many of the services run at the highest access level in the system (Admin) and thus if an attacker can plant a backdoor in such a service, the attacker would have almost unlimited access to manipulate system settings, add new accounts, shut down security mechanisms, and other dangerous activities. Aside from having hackers invade your systems, having an ill-intentioned employee gain unauthorized access to an administrative user account is also a security risk. Most of us want to believe the best about our co-workers, but a surprising percentage of automation system compromises have been due to malicious (or at least unauthorized) insider activities. Managing and controlling who has user accounts on various computers, and the role-based access rights permitted to those accounts, is an important cyber security measure. Even more important is controlling and limiting who has administrative access rights. With distributed architectures that may incorporate a large number of PCs and servers, it is becoming more common to see centralized access authorization and authentication implemented. A Microsoft Active Directory (AD) domain controller can be used to control and manage user accounts and access rights on all of the computers in your SCADA system, as well as pushing-out updates, security policy settings, and account changes. But not everyone needs a user account on all of the computers in a system, and some (like operators) don't need a user account on any computer, depending on how the system controls operator access. In most SCADA operations, the use of administrative-level system utilities and functions should be restricted to the smallest possible number of qualified personnel. It is also important to remove accounts when no longer required, such as when a person leaves the organization (especially if terminated for cause!).

The day-to-day operations of a SCADA system rarely require the use of these operating system utilities. In fact, one way to identify tampering with a SCADA system is to monitor the use of administrative accounts and activities by checking the system logs. We will be discussing tools that perform automated log analysis in a later chapter. When a SCADA system is fully operational and not undergoing a software or hardware update or an expansion, most operating system

utilities have little purpose in normal operations. For routine, periodic system administration purposes, some of these utilities may be needed. The same is true if system-level or application programming is taking place. Otherwise, these programming tools and utilities would not typically be used. Table 4–1 describes typical system utilities, their function, and the personnel who would normally need access to such utilities.

Table 4–1. Standard operating system administrative utilities

Utility	Description/Misuse	Valid Access By
User account management	Allows adding, deleting, and modifying user accounts and user access privileges. User can create new accounts and grant administrative rights to a user	System administrator
Command line interpreter	Like the DOS tool in Windows. Provides command access to other system utilities and functions, generally at the "root" (system administrator) level	System administrator System programmer
File/storage manager	Permits listing of files, creation of files and directories, copy and movement of files, deletion of files, modification of file protection settings, transfer or copy of file to removable media	System administrator System programmer
Task scheduler	Controls the time at which various programs are run and the frequency. Scheduling can be revised, postponed, or cancelled. New tasks can be added to the queue. If event-triggering supported, then trigger causes can be modified or disabled	System administrator System programmer
Task manager	Displays running, blocked, and paused tasks, allows priority changes, reallocation of CPU, blocking of tasks, elevation of task's access rights	System administrator
Network utilities (ftp, telnet, ping, etc.)	Used to transfer files between computers, gain remote access to the command line interpreter and to diagnose networking problems. Could be used to probe for vulnerabilities, to replace files, to "map" a remote network and to misuse remote utilities	System administrator System programmer Network administrator
Network management utilities	Assign IP addresses and computer names, work group and domain names, network sharing of resources, peripherals, and files, remote access rights, firewall administration, router administration, enable/disable ports and services	System administrator Network administrator
Peripheral configuration utilities	Set up parameters for various peripherals, define naming conventions, assign backup peripherals, enable and disable peripherals, modify their operation	System administrator

Utility	Description/Misuse	Valid Access By
Relational database management	Create accounts and access rights to databases and tables, maintain tables, delete or empty tables, modify table structures, add/delete table entries	System administrator Database administrator System programmer
Printer management	Assign printers to given applications, groups, establish buffering and backup assignments, enable and disable printing	System administrator
Programming tools	Editors, compilers, assemblers, interpreters, debuggers used to develop/modify test and deploy programs that will run on the system	System programmer
Scripting tools	Interpreters that will execute a script such as a PowerShell, DOS Batch file, VBscript, Perl, or JavaScript file	System programmer System Administrator
Backup management	Defines where, when, and what to copy to backup media, maintains audit trail of backups, controls mode of backup (full/incremental) and backup verification, controls number of backup versions and deletion of old backups	System administrator
Text editor	Examine file contents, creation of new Batch files, editing of files, modify system setting/ini/config files, modify configuration tables, modify script files	System administrator System programmer Network administrator Database administrator

Because operating systems have differences, the utility programs are described in a generic manner, nonspecific to any particular operating system. Also, system tools used for program development and maintenance have been separated. These tools and utilities are addressed later in this section. SCADA system vendors generally rely on the security features built into the commercial hardware and operating systems they use as the basis for their systems. They may add additional layers of access control (for the SCADA portions of the system, particularly the Engineering and Operator functions). However, they generally rely on commercial security solutions and integral security features for their basic operating system and networking security. Some SCADA vendors will integrate third-party cybersecurity measures into their product offering on an optional basis, but your own IT department can probably do this just as well and possibly for less money.

In a later section, cybersecurity technologies, including *intrusion-detection systems* (IDSs), will be discussed in detail. Essentially, an IDS is like a burglar alarm for a computer and/or computer network. One way to monitor for intrusions (or irregular use of the system utilities) is by learning the typical ways that various system software is used: by whom, with what frequency, for how long, using how much memory and CPU time, on what days, and so forth. When setting up an IDS (specifically a Host-based IDS or HIDS), it is important to develop a baseline that

can be used to identify statistically significant operational differences. Noting that a program is suddenly doing things it never did previously (e.g., sending outgoing IP messages to another computer, accessing a critical system file, etc.) is a form of *anomaly detection*, but it requires knowing what is the normal program behavior. An IDS can spot such activity and alert the system administrator, who can take the necessary steps to eliminate the infestation, possibly by reloading some of the system from backups.

SCADA System Utilities

Just as a basic operating system will come with a set of standard utilities, services, and applications, a basic SCADA system will include specialized utilities necessary for SCADA system configuration, maintenance, modification, and expansion (not for normal operational use). These are utilities specific to the SCADA system vendor and particular SCADA software package(s). While a generic hacker will probably not have familiarity with these utilities (or with SCADA systems), a terrorist (or a disaffected ex-employee) who intends to target a specific SCADA system can be assumed to have access to the necessary documentation and even vendor training or personnel with practical experience with this specific model of SCADA system (and they can always buy one). Keeping track of who uses what SCADA system utilities is just as important as tracking and controlling the use of operating system utilities. Not everyone needs access to these utilities, or has the proper training to use them, and their use should be restricted. Typically, these specialized tools are used by SCADA system engineers and technicians who have received specialized vendor training. Some SCADA facilities train operators to build/edit custom graphic displays so that they can setup operational displays specific to their preferences. But most operations do not allow operators to make database changes or system configuration changes as those are considered as engineering activities.

Most SCADA systems include an additional level of operational access control that is incremental to, and separate from, that which is provided by the computer operating system. A SCADA system operator may not need a user account for the underlying operating system, but they may need one for access to the operational displays. A system may have differing levels of operator access: a view-only mode for training purposes and a full access mode for authorized senior operational personnel. Access to SCADA system utilities and configuration tools may—or may not—require this additional level of authorization. Some less sophisticated SCADA systems merely rely on the computer operating system passwords and access controls to restrict access to SCADA utilities. This is a less preferable scheme, because people who may need access to operating system functions often have no reason to utilize SCADA utilities and tools (or adequate training) and vice versa.

It is commonplace for the majority of SCADA system vendors to further restrict access to SCADA utilities by either job function or with specific user

access authority. Thus, a senior system operator (Fred) may be granted access to use the graphical display editor, whereas an operator trainee (Joe) would be denied access to create/edit operator displays. (This might also be managed with a password known to Fred but not given to Joe.) Fred's SCADA system login and password would define his access to SCADA utilities and functions, separate from any operating system access he may have. In fact, Fred may have *no* operating system access rights and yet have full SCADA system utilities access rights. In most SCADA (and DCS) systems, it is common not to require an operator to login to their operator workstations/consoles because of safety concerns. Generally, these workstations/consoles are fully operational and updating at all times. Actions like automatic logout after so much idle time or switching to a screen saver are not typically considered as acceptable or safe for operator consoles. Some systems may support a shift change function of some type so that the system can associate, and tag, subsequent operator actions with Fred's name and not Scotty's. But none of that typically shuts down the operator console/workstation or requires some form of logout/login process.

One of the most important of the SCADA utilities is the user account management utility, which is used to grant access rights to the SCADA system operational and support personnel (fig. 4–3). In most SCADA systems, this sort of utility provides a means for determining the specific set of SCADA system resources and tools available to given operational, supervisory, and engineering personnel and the limits on how those resources can be used. This allows fonctional roles to be defined and assigned appropriate rights, and then specific personnel to be identified as performing that (those) role(s). In that aspect, this utility is much like the account management software of the operating system, except that it specifically deals in SCADA system resources and tools (Windows knows nothing about the assignment and adjustment of *HiHi* and *LoLo* alarm limits on key process measurements, that is a function that is handled within the SCADA software). The assignment of access rights in a large percentage of SCADA systems is along job category lines (a.k.a., *role-based*). A senior operator, for example, will have more access rights than an operator trainee. Often the main purpose for such security is oriented more toward prevention of human errors than protection against a cyber assault. But it supports both purposes.

Table 4–2 lists standard SCADA system utilities, their function, and typical access frequency during normal operations. As was mentioned previously, every SCADA system vendor will have their own utilities, which may be slightly different in name, organization, and function from those given in table 4–2, but it provides a good representation of the basic SCADA utilities found in most, if not all, SCADA systems.

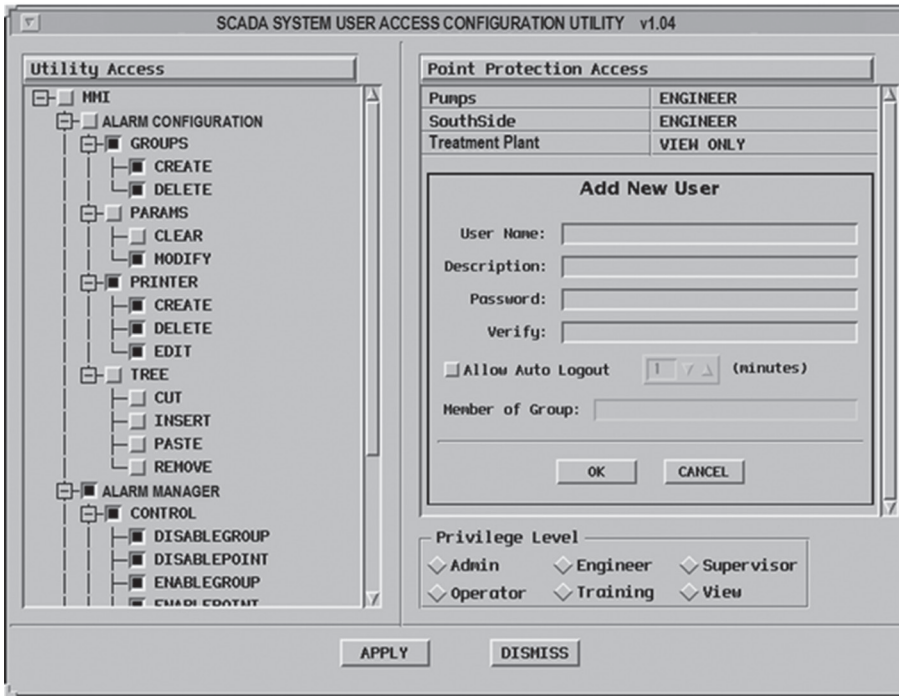


Fig. 4–3. SCADA system user account management utility

Table 4-2. SCADA configuration maintenance and management utilities

Utility Type	Description/Use	Usage Frequency
User account management	Allows the creation of a new user, changing access rights, deleting users	Infrequent, only when personnel changes are made.
Configuration	Modification of physical and/or logical configuration of system, adding or removing RTUs, polling channels, operator consoles	Frequently during initial system commissioning and then again when system expansions or changes are made. Otherwise not used.
Database	Edit, delete, expand the database definitions, space allocations, supported types, logical groupings, access levels	Constantly during initial system commissioning and then again when system expansions or changes are made. Otherwise not used.
RTU downloading	Initiate download of parameters, configuration, initial programming, database information, local calculations, new firmware, local control logic	During initial RTU installation and commissioning and again if RTUs are repaired or replaced. Otherwise not used.
RTU polling	Assign RTU polling priority, add or remove from polling, force a poll, inhibit polling, change polling parameters	During initial RTU installation and commissioning and again if RTUs are repaired or replaced. Otherwise not used.

Utility Type	Description/Use	Usage Frequency
RTU diagnostics	Run tests on RTU, control outputs, read inputs, force tests of RTU HW and communications, read inputs, and control outputs	During initial RTU installation and commissioning and again if RTUs are repaired or replaced. Otherwise not used.
HMI configuration	Set up the semi-automatic and automatic displays, assign displays to groups, set up navigation, assign access requirements	Frequently during initial system commissioning and then again when system expansions or changes are made. Otherwise not used.
HMI custom display editor	Build and modify custom graphics, assign controls, set links to other functions, set up program triggers	Frequently during initial system commissioning and then again when system expansions or changes are made. Otherwise not frequently used.
Operator access management	Define and modify the specific levels of access (run/read/modify) assigned for each user category, utilities allowed, fields and controls allowed for user access, modification	Infrequent. Mainly used for initial system setup and commissioning. Rarely used thereafter.
Operator log	Manual logbook for recording of operational information and for passing important messages and notices between operational shifts. Messages can be entered, listed, printed, removed; messages can be directed to job categories, specific personnel, and can have pre-defined topics for sorting purposes. Messages automatically time-tagged, and operator-tagged on entry	Used constantly during normal operations
Alarm management	Defining and modifying alarm levels and thresholds, defining alarm annunciation options, enable/disable alarming on points and groups, creation of alarm groups, enable or disable alarm notification, assign printers to alarm levels and groups	Frequently during initial system commissioning and then again when system expansions or changes are made. Otherwise not used.
Trending management	Assign values to be collected, remove values from collection, edit archive files, purge archive files, change collection intervals, set storage durations	Frequently during initial system commissioning and then again when system expansions or changes are made. Some functions may be used on a regular basis for system administration purposes.
Reporting package	Configure report formats and content, define calculations, define values to be reported, define report-triggering events and time triggers, remove reports from list	Frequently during initial system commissioning and then again when system expansions or changes are made. Otherwise not frequently used.

Utility Type	Description/Use	Usage Frequency
Computations and calculations	Define, modify, and delete calculated values, adjust calculation parameters, assign re-calculation intervals	Frequently during initial system commissioning and then again when system expansions or changes are made. Otherwise not used.
Logging	Assign alarms and events to logging, remove from logging, print logs, purge logs, assign printing and archive directory to logs, define log retention intervals, define operator and other user actions to be logged, define other activities to be logged, remove activities from logging	Frequently during initial system commissioning and then again when system expansions or changes are made. Some functions may be used on a regular basis for system administration purposes
Data exchange	Set up values and status and controls to be exchanged, assign placement and scaling of values in exchange messages, add or remove points, controls, and functions offered to the exchange partner	Frequently during initial inter-system link commissioning and testing and then again when system expansions or changes are made. Otherwise not used.
Redundancy synchronization	Establish the trigger events for fail over to backup, establish schedule for data update of backup computer, enable or disable fail over function, enable or disable data synchronization functions	Used during the initial commissioning and testing of the system. Not normally used thereafter.

The SCADA functions of a SCADA system product normally provide for a wide range of operational configuration flexibility, since a given vendor's SCADA product needs to be able to accommodate a wide range of application and industry variations. Many of the SCADA utilities exist for the purpose of setting up customer-specific and industry/application-specific configuration tables that direct the actions of the generic SCADA software modules (fig. 4–4). For example, the number of polling channels, the number of RTUs per channel, baud rate per channel, and protocol to be used on each channel are all customer-specific configuration parameters that would need to be defined in order for the SCADA system to perform its RTU polling duties. For a SCADA system of any reasonable size there is a massive amount of information that must be defined to make the system work. A single analog input can require a dozen or more parameters to be set, and a SCADA system might have many thousands of analog inputs. Everything from giving inputs names and descriptions to defining how they are to be displayed and even how they are to be historically trended must be defined.

Configuration activities go well beyond merely setting up the polling channels. It is necessary to describe each and every input and output that the SCADA system is to process for every one of the RTUs. The description for each I/O signal (point) includes assigning a *tag name* to the signal, defining what type of signal it is, providing all of the processing and alarm-checking information necessary to manipulate the point, defining the frequency at which the point is to be processed, the actions to

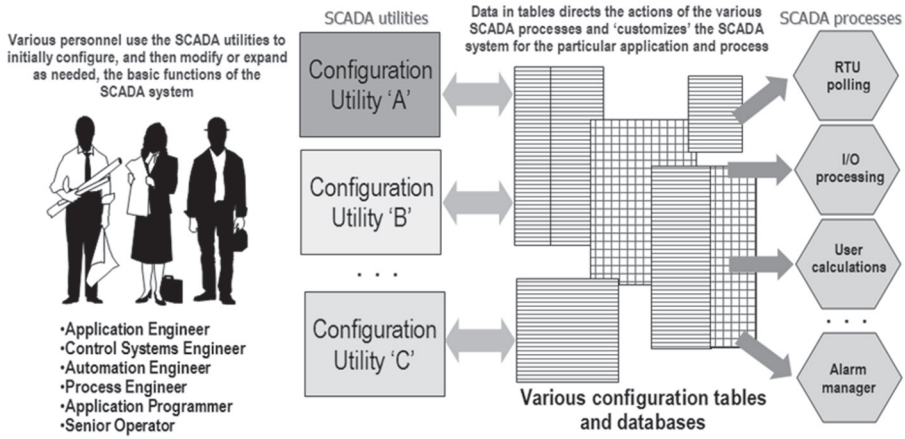


Fig. 4–4. SCADA configuration utilities

take if the point's value exceeds acceptable limits, and many other items. With most SCADA system implementations, the starting point is to identify each location where an RTU will be located and the I/O signals (and local serial interfaces) the RTU will require (and, of course, how you will bring communications to the RTU).

The process of filling in all of the necessary tables with detailed configuration data used to direct the collection and processing of RTU I/O is sometimes called *building the tag/point database.* In addition to the actual physical inputs from the field, SCADA systems also generally compute a large number of other values using the inputs from the field-based RTUs, allow for manually entered parameters, and even collect and perform calculations on historical data. These may be simple or complex numeric calculations (e.g., statistical) or even logical (Boolean) calculations that produce a two-state result: true/false, on/off, safe/unsafe, etc. In many SCADA systems, the point definition database creation utility may also be used to define these pseudo-points (a.k.a. calculated points) whose values are derived by user-defined calculations (fig. 4–5). Most SCADA systems allow for simple value calculations. But if not used to define the calculations themselves, because the calculations are too complex and possibly model-generated, the point definition creation utility may at least be used to create a database placeholder (sometimes called a supervisory point/Tag) for the calculated value and assign alarm checks and display parameters to these pseudo points whose values will be generated separately. The point (or *tag*) database contains information that will be used to direct the host computer's collection and processing of the RTU inputs, and in the case of smart RTUs, some of this configuration information may be downloaded into the RTUs themselves. If any of the logic or calculations needs to be performed in the RTU, then these definitions would need to be sent to the respective RTUs. If an RTU needed to have engineering unit values for local calculations, control, or display purposes, then engineering unit (EGU) conversion factors, and possibly even

alarm limits, would need to be downloaded into the respective RTUs. And finally, if the RTUs are to perform control and sequence logic, this logic will need to be defined and loaded into the respective RTUs. All of this information definition and data entry takes a lot of time and energy and is a large portion of the overall manpower invested for initial system implementation. (Which is also why the resulting data files need to be backed up and stored in a secure manner.) Some SCADA systems and smart RTUs compute a CRC value for each database point (a.k.a. database block) and include that value as part of the point's downloaded information. This allows the RTU to both confirm accurate receipt of the download, but also allows the RTU to perform a memory integrity check, if needed. Obviously, that CRC value only covers the static portion of the point's definition data.

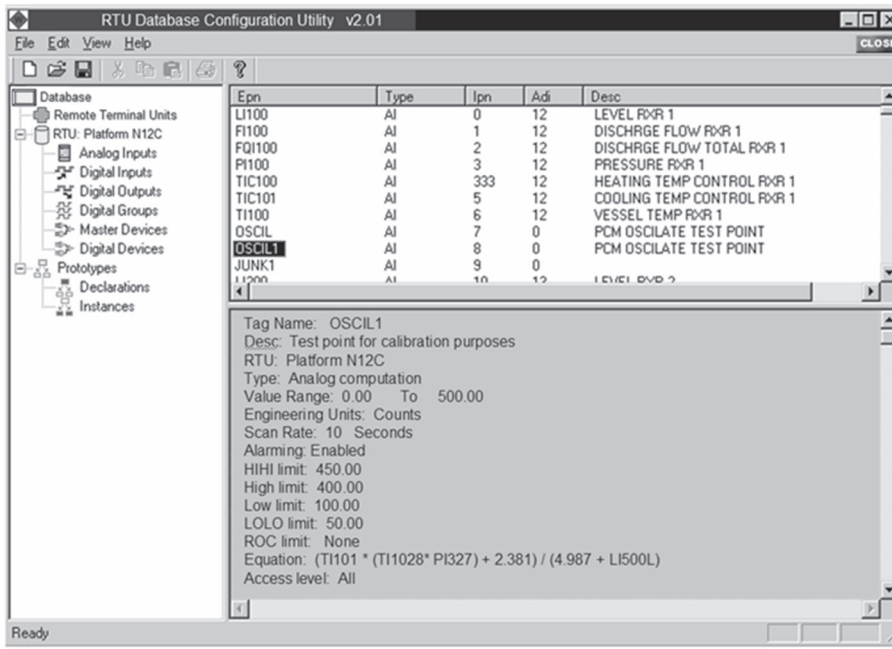
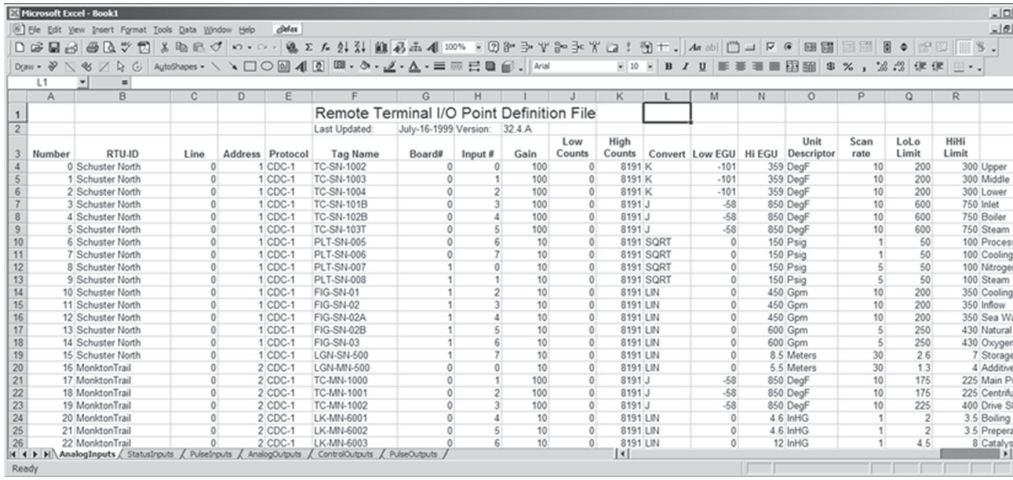


Fig. 4–5. Database point and calculated point creation utility

Many SCADA software packages are designed to take advantage of third-party software such as Microsoft Excel (or the equivalent applications in *Open Office* or *LibreOffice*) and have the ability to both import and export their point tag database using a *CSV file* format. This allows the process of database generation to be performed offline and prior to the staging of the SCADA system (fig 4–6). Aside from being able to use spreadsheets to capture tag definitions, some SCADA packages support various mechanisms for providing live data (e.g., real-time measurements, status) and even historical trend data to Excel spreadsheets so that Excel spreadsheets can be used for generating reports and as operational summary displays.



Microsoft Excel - Book1

Remote Terminal I/O Point Definition File

Last Updated: July-16-1999 Version: 32.4 A

Number	RTU-ID	Line	Address	Protocol	Tag Name	Board#	Input #	Gain	Low Counts	High Counts	Convert	Low EGU	Hi EGU	Unit Descriptor	Scan rate	LoLo Limit	HiHi Limit	
0	Schuster North	0	1	CDC-1	TC-SN-1002	0	0	100	0	8191	K	-101	359	DegF	10	200	300	Upper
1	Schuster North	0	1	CDC-1	TC-SN-1003	0	1	100	0	8191	K	-101	359	DegF	10	200	300	Middle
2	Schuster North	0	1	CDC-1	TC-SN-1004	0	2	100	0	8191	K	-101	359	DegF	10	200	300	Lower
3	Schuster North	0	1	CDC-1	TC-SN-101B	0	3	100	0	8191	J	-58	850	DegF	10	600	750	Inlet
4	Schuster North	0	1	CDC-1	TC-SN-102B	0	4	100	0	8191	J	-58	850	DegF	10	600	750	Boiler
5	Schuster North	0	1	CDC-1	TC-SN-103T	0	5	100	0	8191	J	-58	850	DegF	10	600	750	Steam
6	Schuster North	0	1	CDC-1	PLT-SN-005	0	6	10	0	8191	SQRT	0	150	Psig	1	50	100	Process
7	Schuster North	0	1	CDC-1	PLT-SN-006	0	7	10	0	8191	SQRT	0	150	Psig	1	50	100	Cooling
8	Schuster North	0	1	CDC-1	PLT-SN-007	1	0	10	0	8191	SQRT	0	150	Psig	5	50	100	Nitroger
9	Schuster North	0	1	CDC-1	PLT-SN-008	1	1	10	0	8191	SQRT	0	150	Psig	5	50	100	Steam
10	Schuster North	0	1	CDC-1	FIG-SN-01	1	2	10	0	8191	LIN	0	450	Gpm	10	200	350	Cooling
11	Schuster North	0	1	CDC-1	FIG-SN-02	1	3	10	0	8191	LIN	0	450	Gpm	10	200	350	Inflow
12	Schuster North	0	1	CDC-1	FIG-SN-02A	1	4	10	0	8191	LIN	0	450	Gpm	10	200	350	Sea Wa
13	Schuster North	0	1	CDC-1	FIG-SN-02B	1	5	10	0	8191	LIN	0	600	Gpm	5	250	430	Natural
14	Schuster North	0	1	CDC-1	FIG-SN-03	1	6	10	0	8191	LIN	0	600	Gpm	5	250	430	Oxygen
15	Schuster North	0	1	CDC-1	LGN-SN-500	1	7	10	0	8191	LIN	0	8.5	Meters	30	2.6	7	Storage
16	MonktonTrail	0	2	CDC-1	LGN-MN-500	0	0	10	0	8191	LIN	0	5.5	Meters	30	1.3	4	Additive
17	MonktonTrail	0	2	CDC-1	TC-MN-1000	0	1	100	0	8191	J	-58	850	DegF	10	175	225	Main Pu
18	MonktonTrail	0	2	CDC-1	TC-MN-1001	0	2	100	0	8191	J	-58	850	DegF	10	175	225	Centrifu
19	MonktonTrail	0	2	CDC-1	TC-MN-1002	0	3	100	0	8191	J	-58	850	DegF	10	225	400	Drive St
20	MonktonTrail	0	2	CDC-1	LK-MN-6001	0	4	10	0	8191	LIN	0	4.6	InHG	1	2	3.5	Boiling
21	MonktonTrail	0	2	CDC-1	LK-MN-6002	0	5	10	0	8191	LIN	0	4.6	InHG	1	2	3.5	Prepera
22	MonktonTrail	0	2	CDC-1	LK-MN-6003	0	6	10	0	8191	LIN	0	12	InHG	1	4.5	8	Catalyst

AnalogInputs / StatusInputs / PulseInputs / AnalogOutputs / ControlOutputs / PulseOutputs

Ready

Fig. 4-6. Creating the tag database using a spreadsheet utility

Intentional or accidental corruption of the SCADA system's configuration tables can cause a partial or total loss of a SCADA system's operability. Undetected corruption of or modification to configuration tables can cause invalid data to be presented to operational personnel and supervisory application programs. Most SCADA systems have some ability to check their databases for corruption and errors (although such a check may only be performed if manually initiated). In some instances, this can be done online (while the system is operating), possibly even on a continuous basis. In other instances, an offline diagnostic check may be required (while switching to the redundant SCADA computer). Incorrect (accidental) configuration information has been known to cause malfunction of equipment and even equipment damage (i.e., the wrong contact output turned on causing a valve to close and a pump to run dry and burn up). One of the most important tests performed on a SCADA system, prior to its being put into full production, is a point-by-point verification of each input and output definition, preferably spanning from the actual RTU I/O all the way to the HMI displays. This would have been part of a factory acceptance test (FAT) in the past, but with SCADA operators acting as their own system integrators these days, it may be done during the field commissioning of the RTU equipment.

Many SCADA systems maintain a tracking/audit log of modifications made to the various databases (*if* modifications were made using vendor-supplied SCADA system utilities). Some SCADA systems make use of commercial relational database packages to hold their configuration information (e.g., Oracle, SQLserver, MySQL, etc.). In those cases, it could be possible to examine and modify table entries without using the SCADA system utilities, thus circumventing the modification-tracking process. Remember that these modification-tracking processes were established to aid in catching and correcting human errors, not for the specific

purpose of system cybersecurity. Having a process and procedure for making and preserving a backup copy of critical system files and databases, whenever a configuration change is made, is a good cybersecurity measure. In a worst-case situation, the ability to restore from backup may be your best strategy to counter a cyberattack. Here, we are not talking about operational setting changes, such as adjusting a setpoint or alarm limit. Those setting changes ought to be automatically copied to the redundant host system by the SCADA software itself (and possibly also to a warm/hot-restart image on the hard drive). And since they may be changed based on process conditions and operational status, the values in a backup made a significant time back may not be useful. But all engineering/configuration settings that control the operation of the system, and which would typically never be changed by operational personnel, should be protected and preserved in a suitably protected backup. It would be nice to believe that we can provide perfect cyber security for our SCADA systems, but being realistic you ought to ensure that you are prepared and able to recover from a (partially) successful cyberattack.

Full configuration of a SCADA system involves much more than just building the point database, although that tends to be the logical first step because much of what follows makes use of tag names from the point database for data references. SCADA systems have to present information to human operators, and although most systems support some level of automatically created displays, most SCADA system operational personnel work from customized, graphical displays that must be created using an editor utility (custom graphic builder) provided as part of the SCADA system software. SCADA systems have included graphical editors for creating custom displays since the 1970s, although the graphical quality and sophistication have obviously improved since back then. The graphical display editor of today (fig. 4–7) allows the development of customer/application-specific displays that can be built without any need for programming (although some may require *code snippets* to handle and process user actions). Most of these tools offer libraries of process objects and images and allow display creation using a drag-and-drop approach. Individual display elements (e.g., a pump or valve) can then be selected, and applicable information filled in on a property form. Custom graphical displays can incorporate actual images (e.g., a map as a background), schematic representations (i.e. a piping and instrumentation drawing - P&ID), a variety of drawing elements, and even audio and video clips. Such displays generally present a set of user-selected inputs and outputs, with color coding and animation to indicate potential problems, in addition to the use of graphical elements to convey the process/plant operating conditions. It is common to include operational control elements in these graphic displays. Next to a pump may be a set of pushbuttons that allow the operator to issue RUN and STOP commands to the actual pump in the field. This usually occurs by linking the graphic element (a button) to the database point for the specific RTU and specific contact output, so that “clicking” the button icon results in having a control command sent out to the RTU at the site

to operate the associated contact output. Clearly, it is important that the database linkages be correct so that commands are not sent to the wrong field equipment. One type of well-known attack on a SCADA system occurred in an Iranian uranium enrichment plant (using the now-famous Stuxnet malware), where the malware essentially inserted itself in the connection between the operator display and the field devices. Stuxnet sent commands to the field equipment (centrifuges operated by PLCs) while providing false information to the operator displays (speed is normal). It must be pointed out that the mechanisms that link graphical display elements to control outputs of the RTUs are usually also available to supervisory application programs running on the host computer, meaning that an application program could, in theory, initiate random control actions in the field unless the system has some means of preventing this (such as requiring an operator confirmation on all application-initiated supervisory control requests).

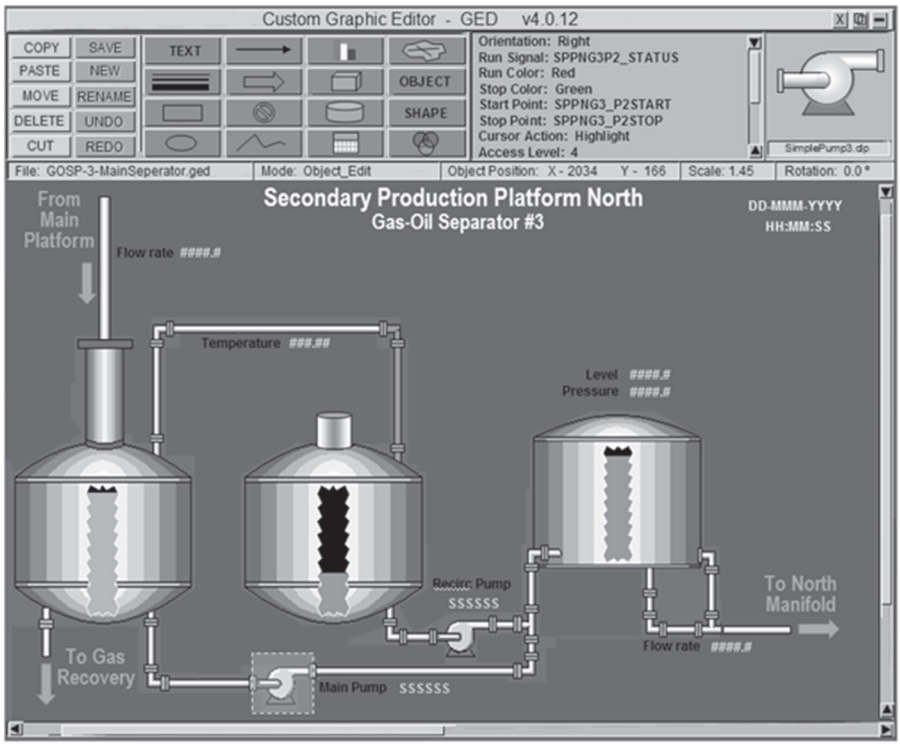


Fig. 4–7. Graphical display editor

In addition to supervisory manual control of field devices, the operator displays may offer access to user-adjustable parameters (e.g., a pressure setpoint) and even to alarming functions (alarming enabled/disabled and adjustment of alarm limit values). One of the security issues relative to SCADA systems is the level at which

access controls are enforced. In some systems, operator access control is enforced within the HMI package. Each graphical display makes checks based on the user currently logged in to an operator's console and based on logic in the HMI package. This means that if someone could access and modify the graphical editor, it would be possible to create a graphic that did not make such checks and then to run that graphical display to gain inappropriate control access. Some SCADA systems place access control information directly into the point database and embed access verification in the underlying database interface level. This prevents maliciously crafted graphical displays (and supervisory application programs) from bypassing access control mechanisms. In many systems, the access to all I/O (and the point database) is managed by a central access library that makes authority checks on the rights of the calling application, regardless of its being a graphical display or a supervisory application optimization routine. Such an access library can also make safety checks (e.g., Is the requested control point tagged?) and reasonability checks (e.g., Invalid to send an analog value to a contact input!). So that access controls cannot be bypassed, it is essential that such a centralized mechanism be in place and itself protected from tampering. Obviously, the tables (access control lists, tag lists, user ID lists, etc.) that such a library uses also need to be protected from tampering. Ensuring this information integrity is a basic cybersecurity concern.

There are many more SCADA system utilities, as can be seen from table 4–2, but we are not going to explore and discuss them all. The important issue regarding SCADA system cybersecurity is to control access to these utilities, track their usage, and employ whatever protective measures are available to restrict their usage to suitably trained and appropriately authorized personnel (whether using account enforcement mechanisms built into the underlying operating system or within the SCADA software itself). From a SCADA systems security design standpoint, enforcing access controls and security policies (for users and application programs) and insuring configuration integrity are primary objectives.

Program Development Tools

Separate from operating system utility programs and SCADA system utility programs are the system and application programming tools included with most SCADA systems. These are generally a combination of programming tools provided by the operating system vendor (compilers, editors, debuggers, etc.) plus additional extensions or software libraries added by the SCADA system vendor. A system may provide general-purpose programming tools, as well as specialized application-oriented programming languages that are unique to a given SCADA system product. For the most part, developing and running user-defined supervisory application programs is a tricky business that requires knowledge of both the operating system and the SCADA system. It is important to ensure

that custom-developed applications don't interfere with the essential functions of the SCADA system. In theory, a badly written program, when allowed to run at a very high priority level, could usurp most or all of the processing power of the CPU and the main memory of the computer, thus effectively shutting down the SCADA system. One reason that most SCADA system owners maintain a separate development system (versus the production system) is so that application programs (as well as configuration changes, operating system patches, and even SCADA software updates) can be tested in a noncritical, non-production environment. Some SCADA system owners actually remove the program development tools and utilities from their production systems so that it is impossible to develop application software on those systems. Unfortunately, most pre-production testing of application programs is done by the same programmer that also wrote the program. This enables the real possibility of having a time bomb or Trojan function embedded within the application code. With the outsourcing of software development to (possibly foreign) third-party foreign contractors, the possibility of having this happen is even greater. Only recently have strides been made in the technology of automatically checking program logic for hidden logic, and even then, most third-party commercial software will never be subjected to such tests (what SCADA system operator has access to such technology?). A procedural strategy to prevent the introduction of malicious applications into production systems is to have a different programmer review the program source code and compile, build, and then verify or test program operation. Breaking up an activity so that multiple people are needed to perform it to completion is called *separation of duties* and is a common practice in financial institutions and the intelligence community. There is an actual software development strategy promoted by the *DHS* (Action/Recommendation 2014), often called a *software assurance program*, that encourages software development and testing procedures that aim to prevent introduction of known vulnerabilities (e.g., C data structures with no bounds checking), bad practices (e.g., hard-coding a password into the program code), and unauthorized functionality (e.g., a backdoor or time bomb) into custom-developed software. For third-party or *COTS* software, there are supply-chain recommendations for working with trusted vendors who, preferably, have their own software assurance program.

Most SCADA vendors have attempted to provide a wide range of configurable utilities that address the vast majority of functions required by their customers, to eliminate or at least minimize the need for custom application programming. This is partially a defensive move, as badly written applications can degrade (or block) the operation of the basic SCADA system. When there is no choice but to address a unique need with custom-developed application software, most SCADA vendors prefer to perform this programming work (and be paid to do so) because of their expertise and knowledge of the inner workings of the SCADA software. Although the migration to standardized operating system platforms has made it possible for third-party vendors to supply certain types of application software, most SCADA

vendors don't like to have software supplied by others running on their systems. (Some vendors will test and validate third-party applications and provide a list of these approved packages.)

The emergence of standardized APIs, such as (classic) OPC, has made it easy for third-party suppliers to produce software that integrates well with various SCADA products. When custom applications must be developed, most SCADA system vendors provide specialized *software libraries* that can be incorporated alongside the user-developed programming and that provide a range of pretested functions that connect the application programs to the SCADA system (fig. 4–8). Some of these functions may ensure that user-developed application programs don't violate access controls or usurp system resources. Some SCADA software package libraries also provide mechanisms for application programs to interact with operator displays (write messages and data values to the screen, pop up a message window, generate a log entry, etc.). In most SCADA systems, the special library functions extend the base set of functions that a programming language such as C/C++ or Visual Basic would include (e.g., file reading/writing, math functions, peripheral device read/write, and date/time functions). These special library functions provide controlled access (and even logging of such access) to various SCADA system components and information. A well-designed SCADA system incorporates security mechanisms that cannot be bypassed, so that the only way for an application program to have access to system resources and facilities is by calling upon the vendor-supplied library. This should be true for both customer-developed and vendor personnel-developed applications. Obviously, the system needs a means for verifying the integrity and authenticity of this library. But this is true for all SCADA system utilities and databases. Most software vendors today provide hash code values that can be used to validate the authenticity and integrity of their various software packages and modules.

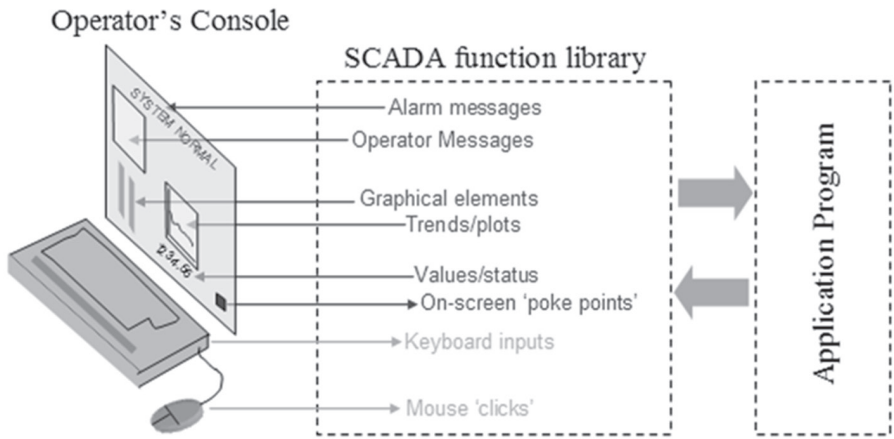


Fig. 4–8. Application program interacting with HMI via SCADA library functions

Some vendors even supply pretested application program templates or scripts that, if used as the basis for custom-written applications, ensure that these applications are well-behaved, clean up prior to exiting, don't leak memory, and run at a suitably low level of priority (relative to critical SCADA system functions). It has generally been assumed, by SCADA vendors and their customers, that all application programs will be developed and then thoroughly tested prior to being deployed, and thus there is no need to make run-time checks on what those application programs are doing. Making run-time checks on applications would add a performance penalty, and, to a degree, that is what an intrusion detection system is for. It watches to see if applications start doing weird and atypical things. (Intrusion detection technology will be discussed in a later chapter.) In order to prevent the running of malicious (or just really badly written) applications, a SCADA system essentially needs to eliminate general-purpose programming tools and replace them with vendor-specific versions that force programmers to adhere to predefined rules and guidelines. One reason that vendors supply special application-oriented languages is that these give the vendor control over what a programmer can, and cannot, do.

Application programs normally have unrestricted access to all components of the SCADA system (especially if written in a language such as Visual Basic). Some of the SCADA system components that could typically be made available to an application programmer, via such a library, are listed in table 4–3. As should be obvious, supervisory application programs normally have a great deal of access to vital system tables and databases. A malicious application program could seriously damage and disrupt the operation of a SCADA system. The process of application program development, testing, and deployment needs to be made as secure as possible with adequate checks and verifications made at key points. The installation of new applications onto a production SCADA system ought to be viewed as a rare and dangerous event during which security and safety procedures need to be followed. Access to operating system-level services and objects (like files) is a separate issue, but application programs ought to be restricted from having direct (non-verified) access to these capabilities. Some SCADA vendors replace standard C/C++ header files and libraries with special versions that inject access verification checks where appropriate.

Scripting languages are another way for SCADA vendors to provide user customization while maintaining control of the system resources and integrity. A SCADA scripting language is a normally a high-level programming language that is specifically tailored to perform supervisory control functions and includes advanced language features that access SCADA system functions and objects in a controlled manner. (Here we are not talking about standard operating system scripting languages such as VBscript, Python, and JavaScript, but rather high-level scripting language created by the SCADA software vendor.) Such languages are called high level because they tend to be functional (what to do) rather than procedural (how to do it), as are low-level languages. A scripting language doesn't

generate actual executable computer program code. It is translated into data that are processed by a vendor-supplied *interpreter* program, allowing the vendor to strictly control both what such a script can (and cannot) do and the resources allocated for (and priority of) script execution. (The same *sandbox* protective/restrictive concept taken with *Java Applets* used in Web pages.) Scripts are often used to perform a long sequence of checks and/or to manage long and involved supervisory control procedures that would normally be done manually and interactively via an operator's console. SCADA scripting languages would be able to reference data objects by their tag name and issue control commands to control objects using their pre-defined methods (e.g., "Send command START to Main_Pump_3"), which would not be possible with a standard VBscript as VB has no knowledge of a 'START command' or what is meant by 'Main_Pump_3'.

Table 4–3. Typical SCADA application programming library functions

SCADA component/Data	Functions
Alarm manager	Read current alarm log entries, append entry to alarm log, read and modify alarm limits on points, and enable/disable alarming on points and groups
Real-time database	Read the current value/status of any input or computed point value. Read, change any parameters or settings in the database tables
Configuration file/table access	Read, modify, delete configuration table entries
Log read/write	Read log entries, append to logs, delete logs, delete entries
HMI interaction	Establish trigger events and data inputs from operational displays, write data to display objects and regions, send messages to operator consoles, adjust the values of displayed data, manipulate graphical objects. Cause pop-up windows to appear, redirect console to alternate display page(s)
Historical data access	Read historical data, append to historical data file
Supervisory control	Send control commands out to RTUs to manipulate analog, pulse, and contact outputs. Modify values of supervisory points. Run/stop local control logic in RTUs
Parameter changing	Read and modify parameters including calculation constants, timers, counters, durations, change settings such as on/off scan, on/off poll

Scripts are often linked to poke points on operational displays, so that they can be initiated by operators through a simple mouse click. They may also be linked to and triggered by alarm conditions and even the system clock and calendar function. (For example, run this script to refill that tank every weekday at 2:00 AM.) The SCADA scripting language restricts the user from getting access to anything other than what the scripting language supports. The vendor's script execution software (the interpreter) ensures that scripts can't run too often or run forever or usurp excessive amounts of memory and CPU time. Nevertheless, most SCADA system

supervisory scripting languages are designed for automatic supervisory control, and thus scripts can reach out to the RTUs and manipulate outputs, making them a danger if not properly validated. (Scripts with unintentional errors have caused enough problems; imagine what an intentionally malicious one could do!)

The development, testing, and installation of supervisory control scripts, onto production SCADA systems, needs to be treated with the same care and cautions as suggested for user-developed application programs. The following example of a supervisory control script demonstrates some of the capabilities of such languages, particularly the ability to effect automatic, supervisory control of field equipment:

```
** Script to bring pump station 81N to maximum pumping state
** Begin by starting all three pumps
ENABLE COMMAND LOGGING
** Set bypass valves for re-circulate mode
COMMAND VLV3N2C4 TO CLOSE
WAIT FOR COMMAND TO COMPLETE OR FAIL IN 60 SECONDS
COMMAND VLV3N11D TO OPEN
WAIT FOR COMMAND TO COMPLETE OR FAIL IN 60 SECONDS
IF NOT FAILED
** Now in recirculate mode, bring up pumps
    COMMAND PUMP3N1 TO RUN
    WAIT TILL PI3N1D EXCEEDS 22.0 PSIG
    COMMAND PUMP3N2 TO RUN
    WAIT TILL PI3N2D EXCEEDS 22.0 PSIG
    COMMAND PUMP3N3 TO RUN
    WAIT TILL PI3N3D EXCEEDS 22.0 PSIG
** All pumps running, now end re-circulation mode
    COMMAND VLV3N2F TO OPEN
    WAIT FOR 60 SECONDS
    COMMAND VLV3N11D TO CLOSE
    WAIT FOR ALL COMMANDS TO COMPLETE
    NOTIFY OPERATOR "Pump Station 3N running at full capacity"
    EXIT

** If both bypass valves did not operate properly
IF FAILED OR TIMEOUT
    NOTIFY OPERATOR "Command timeout on Pump Station 81N"
    END COMMAND LOGGING
    EXIT
```

A malicious supervisory script (or application) could send out commands to operate process equipment in a manner that causes equipment damage, disrupts operations, and even creates a hazard to health or life. Some SCADA system owners require that supervisory control applications and scripts be developed with added steps that call for human confirmation of all control and parameter change functions. In such a scheme, the program/script would reach a point where control was called for and would pop up a message asking the operator for permission. In some cases, the operator is given 'x' seconds to either approve or abort the action, or else the program/script defaults to "approved" and implements the action. This is fine for applications that run infrequently, but if an operator must constantly hit the approved button, it becomes automatic and reflexive and the operator doesn't even look to see what is about to happen. Also, inclusion of operator permission checking is not enforced by technical means, making its inclusion something left to the programmer to incorporate. One way to enhance SCADA security would be to have operator permission checking embedded in the scripting language (and access library) for any controls or parameters designated (during system configuration) as critical. In this way, the system would enforce such an authorization check, regardless of a programmer including it in his program logic. A technology called *white-listing* can be used to prohibit new, not previously authorized, software (including scripts) from being permitted to execute. We will discuss this technology in a later chapter. It is one way to block the possibility of malicious introduction and execution of unauthorized and unapproved supervisory programs and scripts on a SCADA system.

Standardized APIs

When SCADA system vendors started consolidating their products onto standard operating system and hardware platforms in the 1990s, this opened up the possibility of having application software developed by third-party vendors—in the same way that anyone can develop software for a Microsoft Windows/Intel (Wintel) platform. All SCADA systems deal with the collection and movement of both real-time and historical data, including moving these data (bidirectionally) between applications and the places where the SCADA system maintains this data. To manage this activity and make program design more uniform, most SCADA systems include some form of API, which offers a well-defined means for application programmers to request and exchange real-time and historical data.

Originally, vendors devised their own proprietary APIs, since no one else was going to supply software to run on their systems! In the 1990s, a lot of work went into defining standards for APIs on the commercial/business side of computing, and the standards that resulted eventually found their way into the SCADA systems because of their adoption by the operating system vendors. Although there was often a goal of nonpartisanship when defining standards, they all eventually

fell into either the Unix/Linux camp or the Microsoft camp. Today, you will find SCADA systems that incorporate one or more of the following popular APIs.

OPC

OPC (now called *OPC Classic*) is a data-exchange standard that came out of the process control world as a result of efforts to interface PC-based graphical display packages (e.g., the FIX® and Wonderware®) with PLCs and other smart devices. In the beginning, these devices mainly used slow-speed, serial communications to exchange data, often with the well-known Modbus-RTU protocol. This protocol was ill-designed for handling the more complex types of data that were commonly available in such devices and originally did not (yet) support an Ethernet-IP version. OPC defined a more sophisticated set of client/server interface mechanisms so that complex data (data with time and quality tags, sequence of events, alarms, etc.) could be exchanged among systems from different vendors, interconnected by a high-speed Ethernet LAN (fig. 4–9). Unfortunately (except for Microsoft), OPC was built on Windows proprietary mechanisms—DCOM (distributed common object modem) and RPC (remote procedure calls)—which meant that to set up an OPC client and/or server, you needed a computer running a Windows OS. Although OPC originally only handled real-time data values, it was expanded for other data types over time, including historical data and alarms and events.

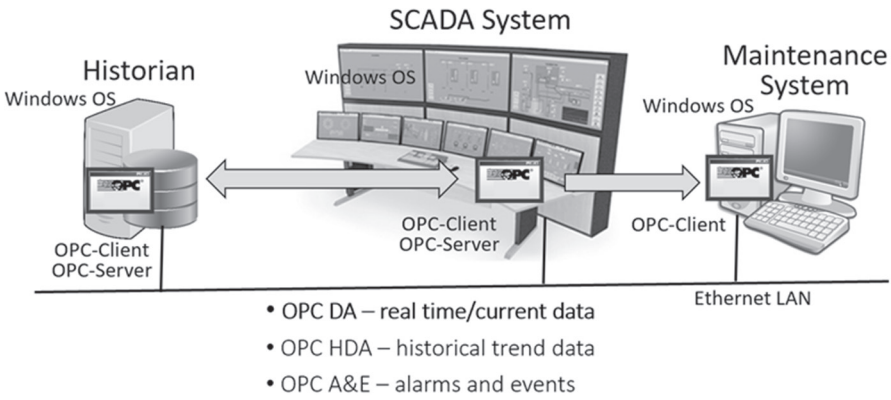


Fig. 4–9. OPC client/server architecture and data-exchange alternatives

Today, several commercially available SCADA systems are built from hardware and software components from multiple manufacturers, with inter-component data exchanges (often over LANs and even WANs) done via OPC. Importantly—and typical of pre-9/11 product and standards development—OPC had no intrinsic security features and has several well-known security vulnerabilities. OPC is an API for a Microsoft Windows operating system environment, as it makes use

of the Microsoft OLE and DCOM features. (OPC vulnerabilities will be discussed in more detail in a later section of the book.) A check of the national vulnerability database will show that DCOM and RPC have a huge number of known exploitable vulnerabilities. Microsoft has actually abandoned DCOM and RPC in favor of its newer network architecture (.NET) and because of this, the OPC committee had to devise a replacement, called OPC-UA—which has NO Microsoft dependencies. Of course, there is a huge installed base of OPC Classic-based systems that do not incorporate any of the new features or security extensions that OPC-UA provides.

OPC-UA

In 2007, the OPC Foundation© launched OPC-Unified Architecture. The Unified Architecture (OPC-UA®) is described in a layered set of specifications, broken into parts. It is purposely described in abstract terms, and only in later/lower parts is it "married" to existing technology on which software can be built. This layering is on purpose and helps isolate changes in OPC-UA from changes in the technology used to implement it. One potential advantage to OPC-UA is that is uses TCP/IP communications and not some Windows-specific, LAN-only communication functions, so it will be usable over WANs and not just LANs, unlike OPC Classic. Also, it is possible to embed OPC-UA communications in non-Windows, network-enabled devices, such as PLCs, RTUs, smart instruments, and other smart devices, so that they can directly exchange data without the need for an OPC gateway device. There is even a manufacturer that has put all of the OPC-UA basic functions onto a chip that can be built into such products. OPC-UA is one of the technologies being evaluated for the IIoT (Industrial Internet of Things).

Standard Query Language

The Standard Query Language (SQL) is not an API per se, but in many SCADA systems, the mechanism for exchanging data with other systems, or between sophisticated application programs, is to pass it through tables in a relational database server (fig. 4–10). Some types of applications, like pipeline leak-detection and line-pack models, involve so much data and configuration information that the applications may use relational tables for their own storage. GIS (graphical information system) applications normally use relational database tables to hold their data as well. This scheme is facilitated through the availability of a platform-independent database query language (i.e., SQL) and the migration of popular SQL-compliant relational database packages onto both the Wintel and Linux/Unix platforms. SQL implementations vary in their level of error checking and validity checking, including user/application credential checking (e.g., passwords), but in general, it can be assumed that current levels of security are insufficient and that there are vulnerabilities that can be exploited to attack a relational database or the applications that interact with and through such databases. One of the threats to SQL-compliant databases

are poorly implemented Web interfaces that provide users with the ability to query their data. If user input checks are not adequate, then maliciously crafted user input (an attack called *SQL Injection*) can actually cause damage to the database tables, their contents, up to and including totally deleting them. SQL-compliant relational database manager packages are available in both the Microsoft world and the Unix/Linux world. The cybersecurity working committee (SP-99) of the International Society for Automation (ISA) recommends creating a DMZ between automation systems (such as a SCADA or DCS system) and other systems that require their data as well as the placement of an RDBMS in that DMZ in order to protect the SCADA or DCS system and eliminate a potential attack pathway. This also protects the SCADA system from excessive data loading in case the data-requesting applications are not well-designed and tested. At worst, they might crash the RDBMS but not the SCADA system. We will discuss DMZ technology in more detail in a later chapter.

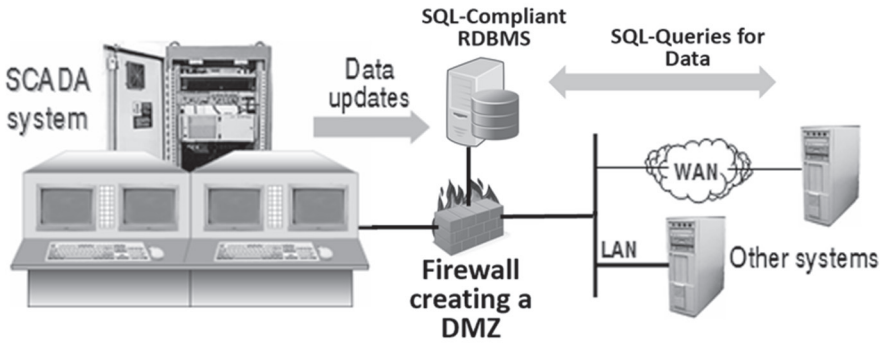


Fig. 4–10. Using a SQL-compliant database server to exchange SCADA data

File sharing with SAMBA

The Unix world was a much more diverse place in the late 1980s and 1990s, as many different computer hardware manufacturers accepted that their own proprietary operating systems were waning in popularity and that a Unix implementation might be a good marketing move. Digital, IBM, and Hewlett-Packard (among others) came out with their own versions of Unix. Although these systems did share a Unix heritage, they were nevertheless incompatible hardware platforms. One of the holy grails of the computing world has been to allow cross-platform, distributed computing, without regard for the make or model of the computers involved. We seem to have reached that point today, but only because all of the computers actually are the same (Intel® x86 family or equivalent). Computers with Linux and Windows operating systems have mechanisms that allow them to share their file systems among a group of computers. SAMBA was developed in the Linux world to allow file sharing with Windows-based systems. Samba is a Windows-compatible file sharing system. It is used to set up Windows share

on Linux systems. Samba is a Linux implementation of the SMB/CIFS protocol. Samba can be used to set up storage server or sharing files and directories on the Linux system so that they appear as a shared drive. Unfortunately, SMB (server message blocks—a message type in the Microsoft NBT protocol) has well-known vulnerabilities, and thus SAMBA file sharing is best used only on an isolated LAN.

Common Object Request Broker Architecture

One effort that came out of work toward achieving information sharing among Unix systems was the Common Object Request Broker Architecture (CORBA). This was a set of cross-platform services and communication mechanisms, as well as an API, that allowed programs running on one computer to request services (look up a value in a database, perform a statistical calculation, etc.) from another computer, without having to deal with the incompatibility issues. (In effect, it was the Unix world's answer to Microsoft DCOM.) Larger, multi-computer SCADA systems, based on Unix variants, have employed CORBA for their data exchange mechanisms. CORBA also lacked security mechanisms, although committee efforts were supposed to address this shortcoming. Originally, CORBA was to be ported to all computing platforms, even Microsoft Windows variants, but Microsoft decided to devise their own, proprietary scheme: DCOM (with some justification).

A particular implementation of cross-application/computer data exchanges using the CORBA approach was being used in the electric power industry. A group of utilities cooperated in creating the Utility Integration Bus (UIB), which is a specific implementation of CORBA with a layer of customization to address the particular types of data found in the various computer systems (including SCADA) used by electric utilities. UIB is a class of middleware that offers publish-and-subscribe mechanisms for sending data only as it changes, from the place where the data is maintained to all applications that need to know when that data changes. Utility Integration Bus (UIB) is a standards-based integration platform designed to significantly reduce the engineering effort required to integrate data in the utility environment. The UIB extends off-the-shelf Enterprise Service Bus (ESB) middleware with utility-specific extensions for support of distributed power system models and standards-based interface services and application programming interfaces (API) using XML messaging per the IEC61970 and IEC61968 standards. The UIB enables you to build a flexible, model-driven architecture for application integration to leverage existing power system-related application investments.

DCOM

Not to be outdone, as CORBA was being promoted in the Unix world, Microsoft fought back with its own version: the Common Object Model, which was for a single computer on which applications needed to communicate; this was followed

by DCOM, which worked across local area networks (not WANs). Once again, DCOM, as with all of the other APIs, did not include inherent security mechanisms. Microsoft, to its credit, has been adding them as quickly as vulnerabilities are uncovered and reported. As was mentioned above, the OPC Classic standard was built upon the DCOM services offered by Microsoft, and so the vulnerabilities in DCOM correspond directly to vulnerabilities in OPC Classic.

ICCP or IEC 60870-6/TASE.2

Ever since the early days of SCADA, the electric power industry has needed to exchange information between local SCADA systems and regional/district control centers. In an effort to standardize these data exchanges, the industry devised ICCP (also called TASE.2 and occasionally UCA1.0) to provide a mechanism for automatic data exchanges. Initially ICCP (inter-control center protocol) was designed to run as an application layer on top of an OSI/ISO seven-layer protocol stack, but in the past 30 years, with the success of TCP/IP, all implementations now use that protocol (IP) as the underlying network stack. ICCP has been extended to support a basic level of authentication (association of control system elements [ACSE]), although this is optional and not always implemented. Because it is normally carried by an underlying IP network, it is possible to use conventional security mechanisms (VPN, link encryption, etc.) to secure an ICCP connection. ICCP uses a client-server design where a system can be either or both in relation to the other system. Although the normal expectation is that ICCP software would be run on the SCADA system, for older systems or systems that don't have the capabilities, it is also possible to implement ICCP in an external gateway/server and communicate with that device in some less complicated manner (such as using a serial, SCADA protocol). ICCP supports a range of data exchange and command/control functions (called "blocks"), but you only need to implement the blocks you need. Part of the initial connection setup is an exchange of information about the blocks supported by each system. Exchanges using ICCP can support the following:

- Periodic System Data- Block1: status values, analog values, quality flags, time stamps, and accumulator/counter values or changes.
- Block 2: report by exception capability for the data types that Block 1 is able to transfer on a periodic basis.
- Block 3: provides a means transferring Block 1 and Block 2 data types as bulk transfers instead of point by point, to improve communication efficiency.
- Block 4: information/operator messages, simple text, and binary files.
- Block 5: device control requests: on/off, trip/close, raise/lower, etc., and setpoints. Includes select-check-operate functionality.
- Block 6: allows an ICCP client to remotely control programs executing on an ICCP server.

- Block 7: extended event reporting to a client of error conditions and device state changes at a server.
- Block 8: generation scheduling, accounting, outage and plant capacity information.
- Block 9: allows a client to request historical trend data between a specified start and end date, from the server.

UCA 2.0 (Utility Communication Architecture)

In the past 20 years, work has gone forward within the electric and water utility industries to define an all-encompassing data and communications architecture that can connect any data source to any user of that data, using an object-oriented model. Although progress in fully defining and promoting this standard has not been without problems, it has made inroads with some utilities and product manufacturers. Various manufacturers of protective relays and substation equipment now offer UCA2.0 communications, and there have been numerous demonstrations of the various components of this rather broad standard. Similar to OPC's ability to give a list of data items and groups in an OPC server, a device that speaks UCA2.0 can provide a list of points/tags that it contains. The types of objects and the data/methods of each are defined in the standard, so there is compatibility among compliant devices and systems. The CASM (Common Application Service Model) specification defines the methods/commands you can issue to various object types. The GOMSFE (Generic Object Models for Substation and Feeder Equipment) specification defines the objects as well as their standardized data-naming conventions (e.g., "PhsAf" for the Phase A floating point value of amperage). Objects are organized into "bricks" with a hierarchy of naming levels within the brick to specify individual data item/element (e.g., the full naming scheme is as follows: Domain/Brick$FunctionalComponent$Object$Element). Within a facility (e.g., a substation or generating facility), UCA2.0 utilizes high-speed (100 Mbps, switched) Ethernet LAN technology, and the messages are built around the MMS standard and protocol. Between facilities and to upper-level supervisory systems, the standard provides for transport over IP networks and even low-bandwidth serial connections. UCA2.0 also suffers from a lack of security mechanisms and is vulnerable to attacks on the underlying communications transport mechanisms. But like ICCP, if it is transported across an IP network, conventional IT security mechanisms can be applied.

Although most of these popular APIs have no intrinsic security mechanisms (as opposed to error detection, correction, and prevention mechanisms), if they are used in distributed system architectures that communicate across a LAN/WAN, it is often possible to tunnel them with VPN technology (for a discussion of VPN technology, see Chapter 3). This does not eliminate the security vulnerabilities present with applications that are co-residing on the same computer (recall the Stuxnet attack) and using one of these data-exchange mechanisms. All such mechanisms

accomplish is to prevent inter-application message traffic to be tampered with or falsified as it traverses the intervening IP network. We will discuss communication integrity and confidentiality in a later chapter.

5

Operator interface

Access-Control Mechanisms

Once a SCADA system is installed, commissioned, and placed into continuous operation, it is primarily the system operators who interact with the system and use the system to monitor and control the target process and field equipment. Other personnel may still need to maintain and administer the SCADA system and perform routine housekeeping and support functions, but it is the operational staff who predominantly access the system on a constant basis.

When you enter the control room of a SCADA system, depending on the industry and the age of the system, you are frequently faced with rows of equipment consoles filled with color displays. In an older SCADA system, these might be CRT displays built into enclosures, but today, they are probably PCs with high-resolution flat-panel displays, possibly with touch screens. You may also see wall-sized informational displays (based on projection video technology or mosaic-tile panel board technology for older systems and just multiple large-screen flat-panel monitors in more modern systems.) You may even find instrument panels with chart/pen recorders, indicator lamps, and manual, push-button controls left over from the pre-SCADA times. Figure 5–1 shows a simple control room design that includes two separate consoles filled with operator displays; a multi-flat-panel video wall, onto which live data from any of the other screens or other data sources can be directed; and a master clock. Depending on the application (natural gas or electric power), there may be a display reserved for national weather and news. The two multi-screen consoles allow two operational groups (or two operators) to deal with different problems simultaneously (and it provides some level of equipment redundancy). I have been in control centers where there were many more than two consoles, but rarely ever have I seen fewer than two. For logging and reporting purposes (and so that hard copy printouts of screen images can be made), the operating consoles each include a set of printer/loggers. There is often a separate computer/server room and telecom room where applicable equipment is located. One reason for the separation is that different people require access to different equipment and areas. Aside from the SCADA system, there may be other computers and workstations associated with engineering and maintenance functions plus the

business applications, but these systems are not directly involved in monitoring and controlling the distributed process via the SCADA system. Of course, there are also small SCADA systems that consist of a desktop PC, a printer, and a master radio, all sitting on a desk in the corner. Pipeline and electric transmission utilities tend to have the big, fancy control rooms. Small water utilities and rural electric cooperatives (RECs) tend to have the desktop SCADA systems.

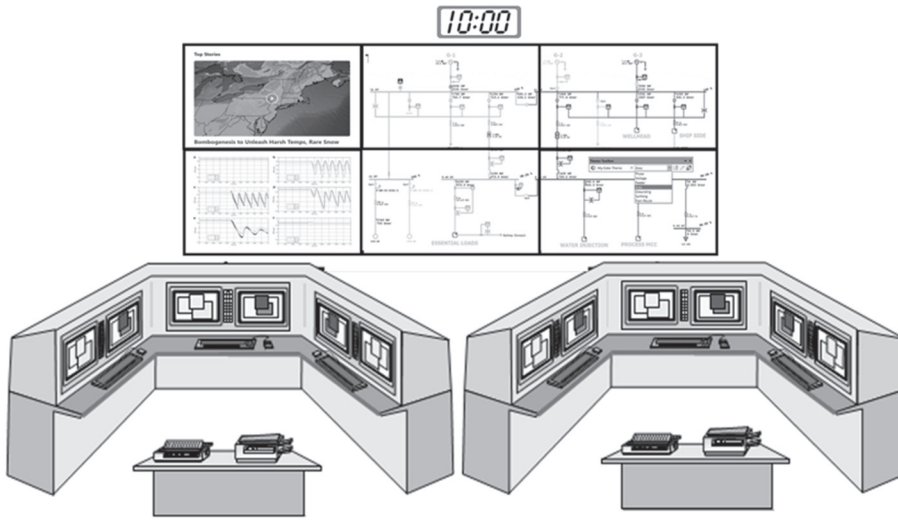


Fig. 5–1. Example SCADA system control room console design

The purpose of all this equipment is to give operational personnel a real-time view of the state of the process they are monitoring and to provide them with a means for initiating control actions and responding to alarms and events, whenever necessary. Obviously, we don't want unauthorized (or untrained) personnel walking into the control room and touching the operator consoles and potentially sending commands out to operate field equipment. So that only authorized personnel have access to the system functions, physical isolation and physical protection of the equipment is common, particularly for the control room (because as we have mentioned, the operator consoles may have no protection other than physical protection because they are always operational and don't require a login or password). As was previously noted, SCADA systems include some type of access-control mechanism for non-operational personnel, whether it is just the user account/password scheme of the underlying operating system or that plus additional SCADA access controls as well. In the vast majority of cases, this is an ID/password scheme, whereby either each user or each category of user is issued a unique ID/password pair that enables (and disables) the functions and features authorized for this user or category of user. The problem with such protective measures is that people don't change or properly protect their passwords, they

log in to system workstations and engineering terminals and never log out, and they even let other people use their accounts; in addition, in the distant past, most vendors build in secret passwords (backdoor accounts) that they could use when customers get locked out of their own systems. Microsoft essentially did this as well for their older versions of Windows. Password protection is better employed for reducing human errors and providing an audit trail of activity than it is for cyber protective purposes. Also, in all computer systems today, the operating system account passwords (users, admin, etc.) are somewhat protected because only their hash is ever stored on the computer. Having said that, note that up to and including Windows XP, the *LAN manager* (a.k.a., LANman) version of user password hashes was also usually stored in the *SAM file* or in Active Directory (unless the "NoLMHash" policy setting was set), and those hashes are easily broken with readily available tools. In some SCADA systems, which have a separate password scheme for system operators or other user logins, those passwords may be stored in an ordinary database table or a text file, making them easy prey for someone with computer skills. Many bad things can happen once an attacker gains access to your network and/or system computers, so we need to do our best to keep them out.

ID/password schemes are used for the purpose of *authentication*—in other words, proving that a person actually is who they claim to be when presenting their ID. ID/password schemes are far too easily foiled, and thus you can't rely on them to ensure that a person using the ID/password of Fred Smith *could be only* Fred and no one else. The basic problem of computer-based authentication is that a computer can only judge by the data provided, it can't see you or recognize your voice (unless you have the right peripherals and software). Recently, some SCADA system vendors have started offering *strong authentication* technology, in the form of multifactor identification schemes, or even using biometric technology. Strong authentication technology will be discussed in detail in the next section. Suffice it to say that strong authentication schemes are aimed at ensuring that Fred must provide proof of being Fred in order for him to gain access to the system, and that no one else can successfully pretend to be Fred. Authentication schemes have taken on a much greater importance with the ability of SCADA systems to support remote users and even operators and with having corporate WAN connectivity. A stranger wandering into a SCADA control room would probably be stopped and questioned, because others would not recognize the intruder. But a stranger coming into the SCADA system electronically, via some remote networking technology, can only be recognized by the authentication process of the SCADA system. If the only mechanism for personnel recognition and access control is just an ID/password scheme, then the SCADA system is potentially far too vulnerable. One way to make remote access more secure is to use VPN technology, not just because this protects the message traffic between the SCADA system and the remote user, but because most VPN technologies require additional authentication in the form of a *USB key/dongle* or a *challenge-response* scheme in addition to the ID and password. Plus, for a remote temporary VPN connection, the user's PC must

have a valid digital certificate installed. Some organizations issue electronic tokens: small electronic devices (or a cell phone App) that generate a 5/6-digit pseudo-random number that changes every 60 seconds and acts as the password. These sorts of multi-factor approaches and mechanisms provide effective user authentication because an attacker would need to steal a token and then use it before the theft was reported. The careful management of user accounts and the use of multi-factor authentication both contribute to the cybersecurity of a SCADA system.

Standard System Displays

All SCADA systems collect real-time and historical information and then provide operational personnel with a wide range of modes in which this information can be displayed and accessed. With most SCADA systems, there are process-related (operational) displays and system-related (diagnostic) displays. We have already discussed the wide range of operating system and SCADA system utility programs that are used to initially configure the SCADA system to perform the necessary tasks. The displays and presentation modes discussed in this section would have been created using those utility programs or automatically created by the SCADA system software. using the point/tag data provided during the configuration process. The operational display functions of a SCADA system are essential and targeting them would essentially blind the operators and block their ability to send control commands to field devices. The well-documented cyberattack on the Iranian enrichment facility essentially consisted of blocking the presentation of valid operational data and providing false data in its place, so that operators were unaware of what was actually happening to the process and plant equipment.

Diagnostic displays

From time to time, it will be necessary to be able to examine the operational performance of the SCADA system and verify that it is functioning properly or to make adjustments or modifications as needed. All SCADA systems provide a set of system diagnostic displays that allow operational and maintenance personnel to check and adjust various aspects of SCADA system operation. A very common SCADA diagnostic display would be an RTU polling channel status display—showing the current polling status of the individual RTUs and the communications robustness of the various polling channels, as well as diagnostic error counters and indicators. Such a display will also often provide a means for altering polling assignments and polling priorities, placing a polling channel in and out of service, and placing RTUs on and off polling. Figure 5–2 gives an example of such an RTU polling and communications diagnostic and configuration display. Typically, such a display would be useful to a system engineer or technician for troubleshooting a communication problem and for adding/removing RTUs in the field. A senior

operator could also use such a display to temporarily prioritize polling of an RTU while trying to resolve a field problem. This example is based on a SCADA system with serial polling channels, but a system with IP networking to the field sites would still need a similar diagnostic display so that at least communication errors could be monitored, and continued connectivity verified.

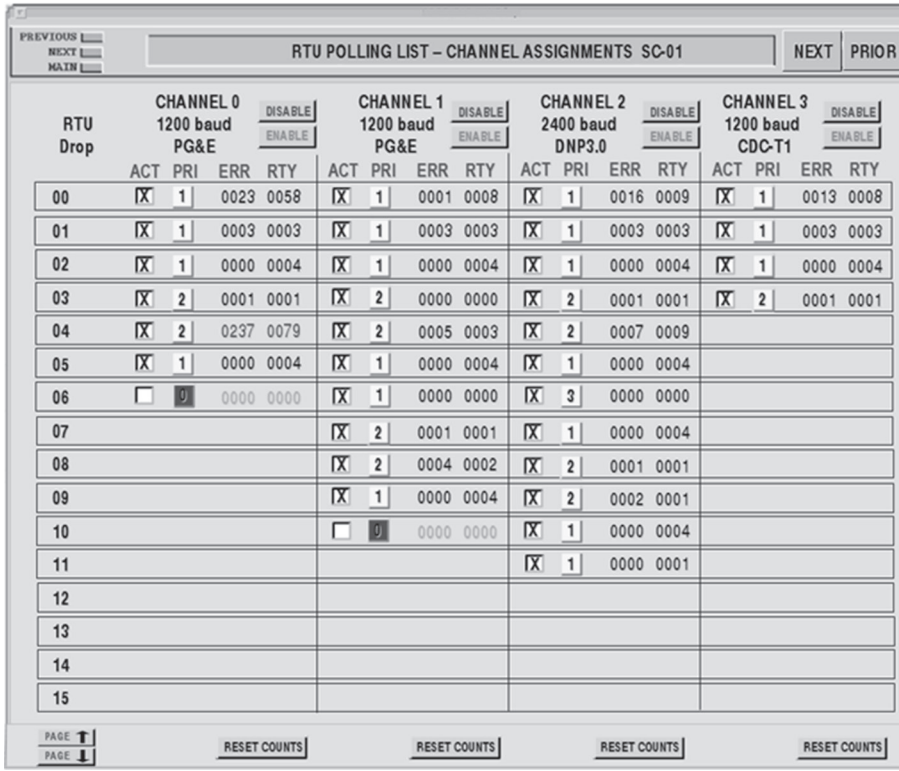


Fig. 5–2. Typical RTU polling and communications diagnostic display

For a system that uses IP networking to the field and RTUs with IP-based protocols, a similar set of diagnostics would be important. Such a SCADA system design might make use of communication statistics collected from network switches, or the RTUs themselves, to populate the display. Performing a periodic "ping" to test and verify connectivity could be part of the information presented. Some RTU and PLC vendors have begun implementing *SNMP* (simple network management protocol) support in their devices which would allow a commercial SNMP client tool to fetch *MIB* (management information base) values and display statistics. Commercial SNMP client software is surprisingly similar to SCADA software in that it allows for data collection from remote devices and offers both standard and user-defined display of that data. Figure 5–3 shows an example of

the sort of display and presentation options available in commercial SNMP client tools. Plotting, trending, and alarming features are all generally standard with most packages. And if SNMP v3 is used, the communications to the field devices are encrypted (but of course you do need to change the factory default *community string* values). Obviously, access to and manipulation of the controls and settings, via such diagnostic displays, need to be controlled through suitable authorization mechanisms. SNMP does give you access to most device configuration settings and so can be dangerous if not implemented properly; SNMP is often used as an attack mechanism because of this. This is why SNMP is usually disabled by default on most PCs and many smart devices and must be intentionally enabled so it can be used.

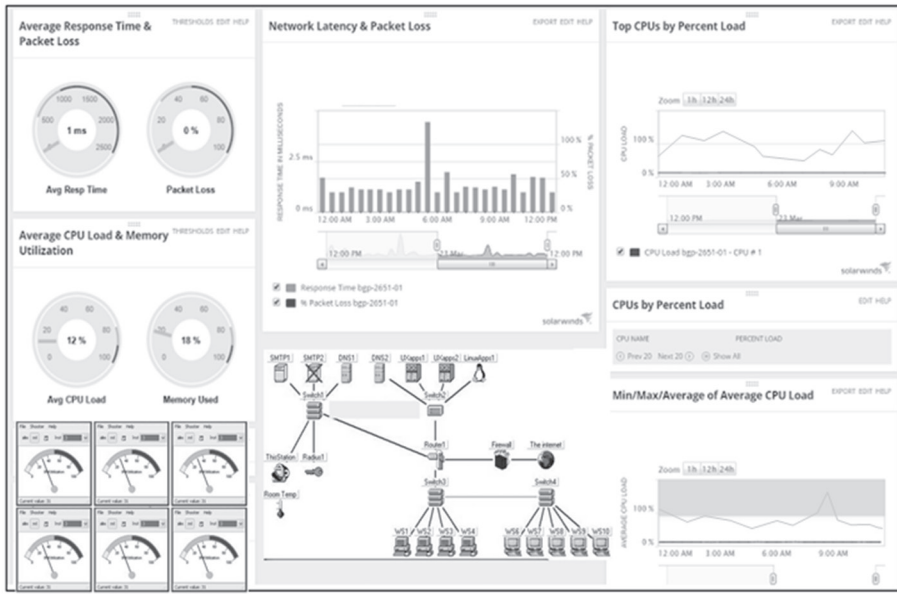


Fig. 5–3. SNMP-based RTU polling and communications diagnostic display

Another typical diagnostic display supplied with most SCADA systems is a system operational status display that shows the gross operational status of the equipment and peripherals that form the system's hardware basis. With some systems, this is an automatically generated tabular display, and with others, a (manually created) graphical pictorial representation is provided, with color encoding and equipment status legends (see fig. 5–4). As with the RTU polling and communication channel diagnostic display, the purpose of this diagnostic display is to offer a straightforward means for operational personnel to identify and isolate system problems. As indicated above, SNMP and SNMP client software can be used for this purpose as well, since all major system components will

be Ethernet-TCP/IP-connected these days, and it can extend to the field devices, if applicable.

Since the vast majority of SCADA systems are built with some level of redundancy, these displays can also provide operational personnel with assurances that the backup equipment is actually functioning—and even with a means for manually initiating an operational transfer to the backup equipment. It has been a common practice to perform software updates to redundant SCADA systems by first loading the new software (after sufficient testing on a nonproduction system) into the backup equipment and then initiating a transfer to this equipment, which then enables the loading of the new software into the former primary equipment. This strategy also allows a return to the old software and equipment, in case the transfer to the new software and equipment is unsuccessful for any reason. As with all such system-level displays, and has been mentioned previously, access to and manipulation of the controls and settings in such diagnostic displays need to be controlled through suitable authorization mechanisms. In a newer SCADA system, as mentioned above, this kind of display could also be generated using SNMP data collection from all of the various components and a commercial SNMP client software package (refer to fig. 5–3), because almost every form of network-connected device, from printers and computers to UPS systems and PLCs now support SNMP and have MIBs from which operational statistics can be extracted using commercial software products. Note that SNMP access is bi-directional (it supports both GET [read] and SET [write] messages) and is typically protected by the use of message

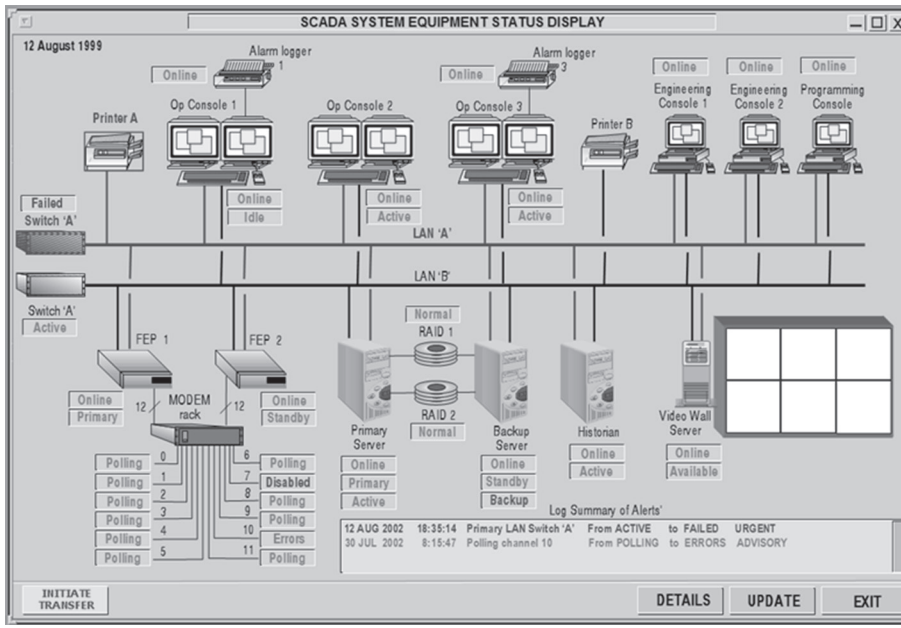


Fig. 5–4. SCADA system operational status display

encryption (in v3) and community strings (passwords). An adversary who had network access could use SNMP tools to attack system components, which would be made far more difficult as long as encryption was enabled and the community strings were changed from their default values ("Public" and "Private") to something stronger (harder to guess).

All SCADA systems incorporate other types of standard (automatically created) displays, as well as these types of displays require little to no configuration effort. A good example of another standard display that might be automatically created would be an RTU current value display. Such a display provides a tabular list of the current I/O values for any selected RTU, and possibly additional RTU diagnostic information, depending on the sophistication of the particular RTU. An operator can usually select an RTU (from an automatically generated list or via an on-screen poke point in some other display) and get an immediate RTU value status display. In some systems, the operator can demand an immediate *integrity scan* of the selected RTU, thus guaranteeing that the data values are current. Figure 5–5 shows what a typical RTU summary display might look like (there would be little difference if the RTU used an IP-based protocol or a serial protocol). Such a display is useful for initially commissioning an RTU and for diagnosing RTU problems. An operator can issue control commands to field equipment via such a display, but it would not be their typical means of performing that function.

Other types of automatically generated standard displays include alarm summaries and equipment tagging lists, as well as displays that list available historic trend pages, custom graphical displays, and log/report pages. Semi-automatically generated displays (requiring a minimal amount of configuration) include point group displays (tabular, bar graph, etc.), where a user must merely define the points/tags to be in each group and possibly a group descriptive heading. These automatically generated and semi-automatically generated displays often compose the largest percentage of the overall display pages available to system users. Since SCADA systems supplanted older telemetry technology in which numeric values were often presented as a voltage signal into an actual panel meter, it was (and still is) common to simulate the same type of informational display on operator displays. Figure 5–6 shows a meter-like (vertical bar graph) point group display of eight (8) related field measurements. In most SCADA systems, it is possible to create groups of four, eight, sixteen, and even thirty-two associated measurements (and to combine actual, computed, and manually entered values in the same group). SCADA vendors have explored a multitude of presentational formats for offering information to operational personnel; meters, bar graphs, slider controls, line plots, and textual presentation formats are normally available with any SCADA system. A Japanese vendor once explored a novel means for indicating the alarm/severity level of a measurement by means of using happy face icons where the facial presentation went from happy to very upset as the alarm and severity level of a condition increased. This idea never caught on with

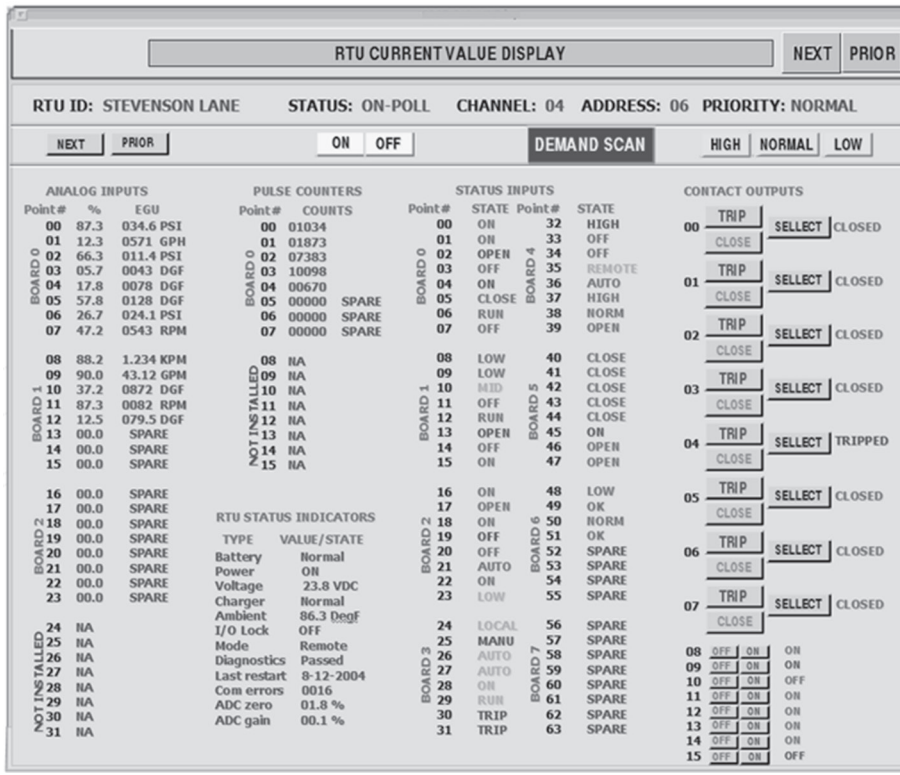


Fig. 5–5. RTU current value display

SCADA system owners but has been adopted by text message users (the so-called emoticon) and in many doctor's offices when asking you to determine how much pain you are feeling.

Two other important types of (semi)automatically generated displays are historical trending displays and alarm summary/information displays. These displays are sufficiently important to justify a lengthier discussion, which follows later in this chapter.

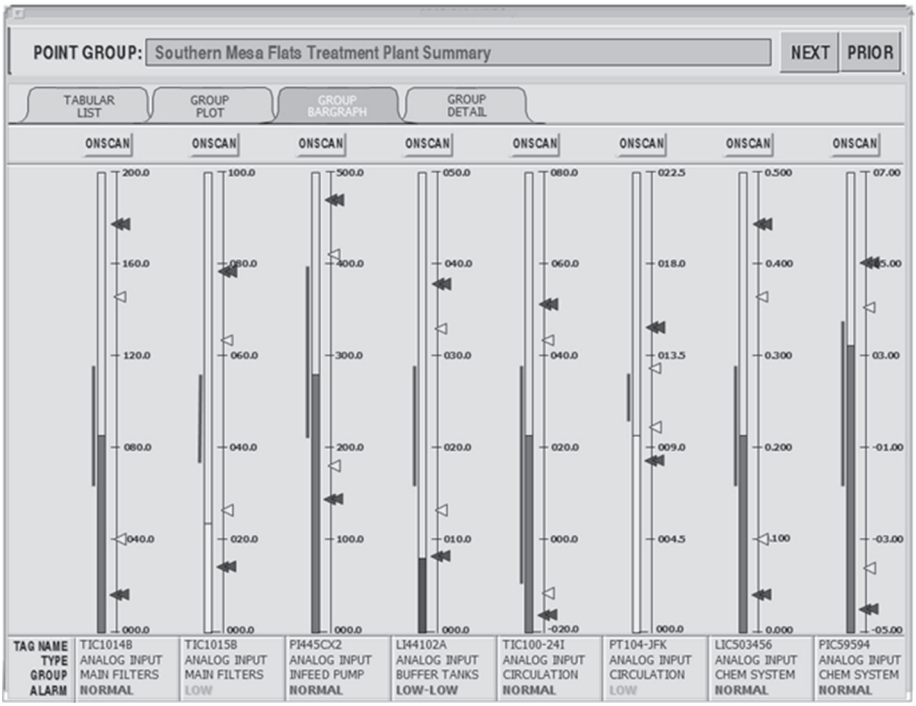


Fig. 5–6. Point group display (bar graph mode)

Site/Industry–Specific Displays

Most SCADA systems (since the 1980s) have used semi-graphical and now fully graphical custom-developed displays as the primary way of presenting information to the system operators. These displays typically require a good deal of time and effort to design, create, edit, and test. Creation of these displays uses a special (vendor-specific) graphical editor utility (one of the SCADA system utilities discussed in Chapter 4) that offers the ability to assemble graphical elements, dynamic data elements, control elements, computed values, and other components into a user-defined display. Depending on the industry and application, these user-defined display pages may be in the form of process-flow diagrams, map displays, or plant/process layout displays.

The ease of creation and editing will vary greatly between products from different vendors, but all graphical editors since about 1990 have supported drag and drop of graphical elements from a menu or table. The drawing capabilities found in a PC package like Microsoft PowerPoint™ or Corel Draw™ are similar to those found today in most current SCADA system graphic display editing packages. Most customized graphical editor packages for SCADA systems allow

importation of standard graphic images in a standard file format such as JPEG or TIFF. These images may be used as *wallpaper* on which other dynamic and static drawing elements can be placed. Most editors also allow the attachment of audio files and video files that can be triggered to play either automatically or via manual initiation. Many also allow for the embedding of sub-windows, which can be linked to video sources or even a Web server (preferably internal to the corporation). Some graphic display editors actually offer the option of creating Web pages so that operator displays can be accessible to personnel with a computer, network access, and a Web browser. Since graphical displays are used to present SCADA system information, the graphical editor must include the ability to select data points and present them, in some user-selected form, on the display page. Numeric data may be presented as a textual numeric value, a bar graph, a meter display, a pie chart, or a trend plot—or even used to alter the physical placement and/or color of other graphical elements. Status information can also be presented as textual descriptions, multistate objects, or color changes, or it can even be used to make other graphical elements appear or disappear from the display. The most popular Microsoft Windows-based SCADA packages include custom graphic editors that support Visual Basic (VB) scripts being attached to specified graphical elements and controls. VB scripts can be used to add animation, user interactivity, input validity checking, and display flexibility to custom graphic displays. Because VB scripts can potentially interact with system settings, controls, and resources, they need to be treated much like a supervisory control program and validated prior to being placed into a production system. The VB scripts built into a user graphic display potentially open up another security vulnerability that could be exploited since Microsoft has given tremendous system access to Visual Basic programs and VB scripts.

Since graphic displays are often required to enable operator supervisory control actions, the graphical editor must offer mechanisms for selecting control points and issuing valid control commands. The graphical editor must also tie in with the operator access control mechanisms to ensure that control actions can be performed by suitably authorized operational personnel only. Via a custom graphic display, an operator may be able to start and stop process equipment, set and adjust regulatory setpoints, modify alarm limit values, and acknowledge alarms. In most, if not all, SCADA systems today, there are really two parts to user-defined graphical displays. The first part is the graphic editor, which provides for graphic design, editing, and modification. This editor actually builds a data file that tells the second part (the part that renders the display) what to draw, much like an HTML file tells a Web browser how to draw a Web page. The second part is the graphical display execution package, which actually draws and updates the displays and interacts with the user. This display execution package may perform the access-control checks when a user requests a display or attempts to initiate a supervisory control action or alter a protected system or operational parameter. Because access checking is

done in the vendor-supplied graphic execution program (which treats the graphic merely as data), there is no way for a graphic to bypass these checks.

Graphic editors that create Web pages, and SCADA systems that provide operational access using Web browsers, open up a potentially exploitable vulnerability because any Web browser would be able to display these operator displays and no SCADA system-specific software would be required. This approach to operator displays is useful when there may be a need for personnel located in another facility to have occasional access to operational displays and because all display definitions would be retained on the Web server and not loaded into individual operator workstations. Figure 5–7 shows an example of a Web-based operational display that provides real-time process information but also allows operator control and manipulation of plant equipment. If Web-based operator displays are used in a SCADA system, then care must be taken to ensure that they are designed with some level of protection that blocks unauthorized use. (The SSL security function described in a prior chapter does allow for mutual authentication using digital certificates and could, if implemented properly, potentially restrict access to only authorized web browser clients.) Some SCADA users only use Web-based operator displays in display-only mode and do not incorporate control elements in those displays. But as figure 5–7 indicates, the decision is left to you. Using Web-based operator displays is far less of a security issue in a SCADA system that is isolated such that an attacker could not gain cross-network access to the system in order to reach the Web server offering the operational displays. Note that regardless of whether the SCADA system uses Web-based operator displays or custom graphical displays, the displays themselves are just data files until "rendered" by a Web browser or a graphic display application. If an attacker can delete or corrupt the display files, then the operator will be unable to monitor and control the process. Protecting these display files (with file and directory access control settings that limit read/write/execute privileges) and maintaining file backups is important for SCADA system cybersecurity.

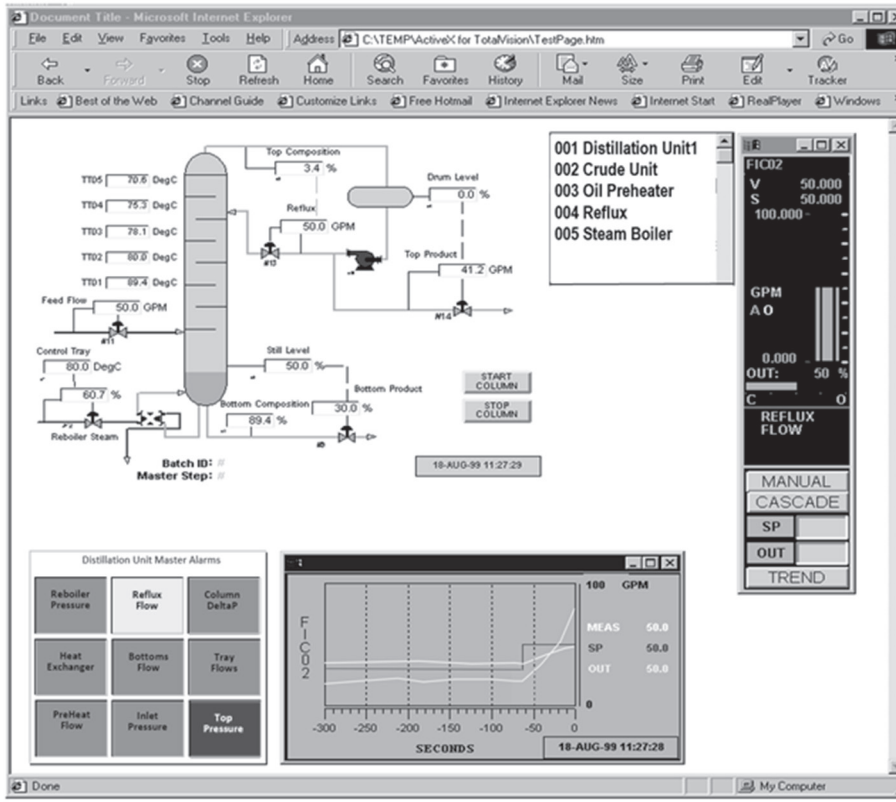


Fig. 5–7. Web-page operational display

Graphical displays

As was already mentioned, most modern SCADA systems make extensive use of graphical data presentation technologies. Graphical data presentation—particularly when there is a physical, geographic, or process-flow relationship—tends to be the clearest and least ambiguous way to deliver information to operational personnel. User-defined graphic displays begin as a blank page onto which a user decides what to place and where and how the display will interact (if at all) with the user. There are many ways to present individual data elements graphically. A custom graphical display provides a means for organizing and presenting a lot of data in a way that an operator can immediately comprehend. Since most applications that utilize SCADA technology are geographically dispersed, a common graphical display strategy is to offer data arranged in a geographically aligned manner. Figure 5–8 shows an example of a geographic display, using an actual service area map as the wallpaper, so that SCADA system data can be directly associated with the physical locations from where it is being collected.

For long-haul pipelines and electrical power transmission, where the "process" is extended over a very large geographic area, it is common to either create a huge display that covers the entire area and allows the operator *to pan and zoom* around that display, or to create a series of displays with contiguous geographic associations and to provide navigation click-points on each to allow the operator to go north/south/east/west of the current area view. Some SCADA systems integrate geographic information system (GIS) technology into their operator displays and add overlays/layers that incorporate field measurements and equipment. This involves a significant database to hold all of the required information, but many of the organizations that take this approach had a need for a GIS regardless of their SCADA system.

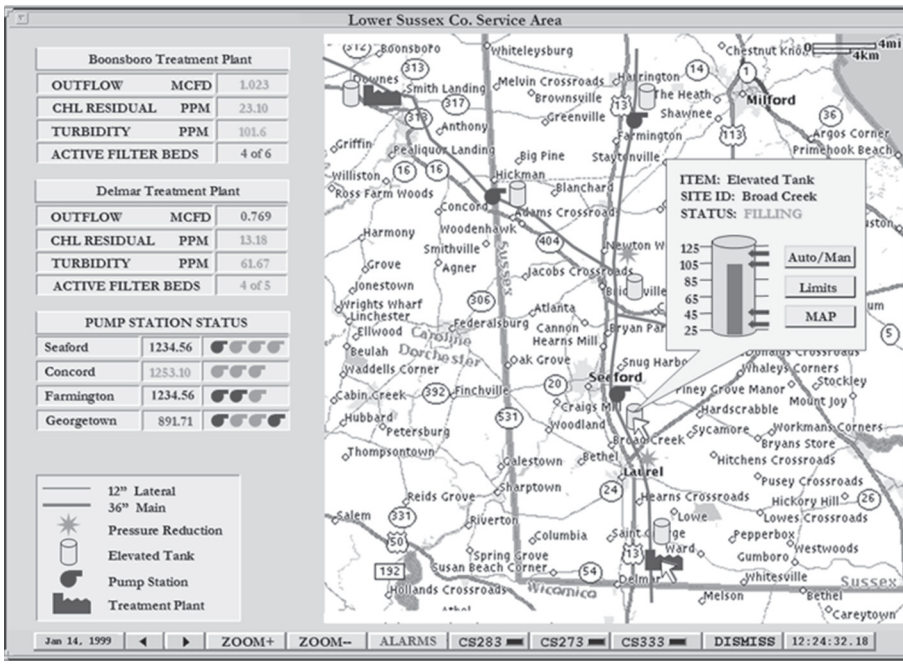


Fig. 5–8. Geographic layout operational display

The pipeline, transportation, and water/wastewater industries tend to make extensive use of geographic displays, and the electric utility industry does as well, to a lesser degree. Often such displays are part of a hierarchy of displays that permit a broad, all-encompassing informational view with *drill-down* (a.k.a. zoom-in) capability that takes operational personnel to whatever level of detail is required. For example, an entire pipeline could be displayed on such a display, and using poke points, an operator could rapidly zoom in to a specific pump station and a specific control point or loop in that pump station.

Another commonly employed graphical presentation design is the process-flow diagram (a.k.a. a piping and instrumentation diagram), which provides a simplified physical interconnection drawing. Initially, these diagrams tended to be stick figure drawings. But today, with libraries of graphical images and the ability to import actual photographs of equipment, such process-flow diagrams are far more realistic and interesting. Process-flow diagrams tend to be used to display specific equipment sets or facilities within the overall process being monitored by the SCADA system. An operator might watch a high-level *overview* display that is geographic in nature and then *zoom-in* to a more detailed process-flow graphic that shows the specifics of the selected substation, pipeline pump station, or water-pumping/storage facility. Figure 5–9 shows a simple example of a multi-window process graphical display that is designed to represent the physical process and equipment for superior operator clarity and comprehension. This figure could represent the lowest level of display drill-down for an operator watching over a large oil/gas offshore production field. Going from a field-wide overview display to the platform-specific display shown in figure 5–9 might only require one or two mouse clicks. It is essential in the design of the operational displays that an operator can rapidly go to a display level that affords the desired or necessary level of information detail. This is essential with SCADA systems that have lots of display pages as the operators need to be able to navigate rapidly and easily.

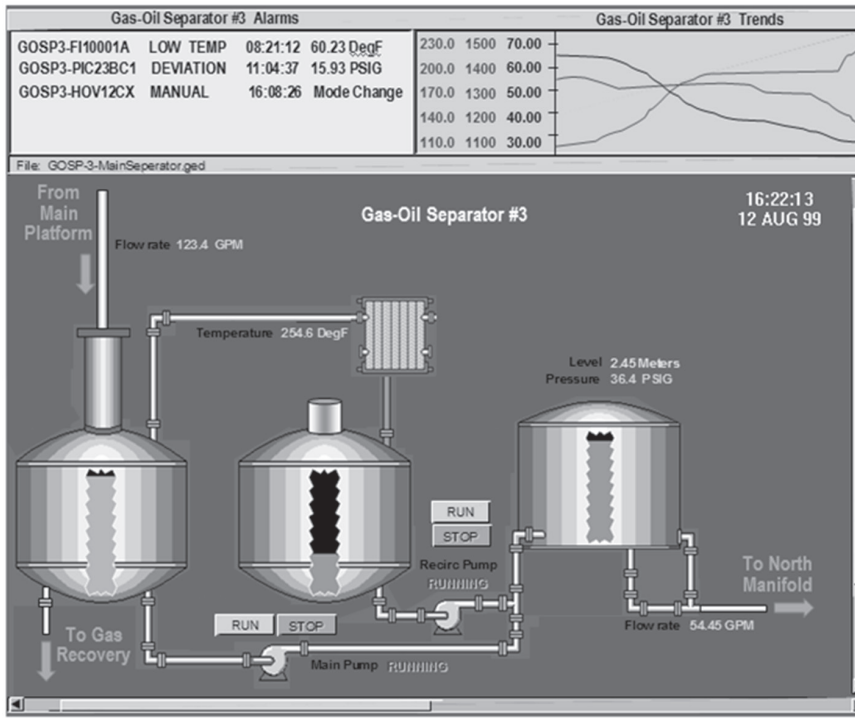


Fig. 5–9. Process-flow operational graphical display

Display hierarchy

In most SCADA systems, there will be hundreds, to even thousands, of display pages available to the operational personnel. Even the best operator can't remember and mentally organize that many pages, let alone the naming scheme used to uniquely identify each. For this reason, all SCADA systems provide some method for quickly and efficiently locating and navigating between these display pages. In the discussion about control room design, one point that may not have been clear is that the operator's only view of the process may be whatever the SCADA system provides. Prior to SCADA, the operator likely would have had a wall-sized map board or mimic panel that actually displayed the entire process (e.g., pipeline, power grid, water distribution system) with basic information and simple alarm indications. A well-designed SCADA operational HMI should not give the operator less visibility. Most operator consoles provide multiple screens so that operators can have several different displays from which to monitor the process. And the HMI should provide a means for rapid navigation to the area of concern.

One commonly used approach to provide rapid navigation is to build displays into an interconnected hierarchy, or tree structure, so that any page can be found by starting at the top of the hierarchy (the root of the tree) and following linkages down or up to the desired page. Such a hierarchy will often begin at a home page that is an overview of the entire process in a condensed form (somewhat akin to the old mimic panel), but with indications of any problems detected by the SCADA system. This home page will incorporate links to logically adjacent or logically dependent display pages. Today, everyone is familiar with inter-Web page navigating on the World Wide Web using hyperlinks. A SCADA system will have a similar set of inter-display linkages. Some SCADA systems now offer their displays in the form of Web pages, thus allowing Web browser and hyperlink technology to be employed. Other systems with non-Web-based operational interfaces employ similar navigational schemes. Navigation may be accomplished by using poke points on displays that transfer to other displays, pull-down menus of available pages, or, for geographically associated pages, a north/south/east/west navigation control (see fig. 5–10) that calls up the display page that corresponds to the geographically adjacent area. One use for this type of navigational control would be to follow a pipeline or a power line across a series of map pages to the desired location. In most SCADA systems, part of the task of setting up the display pages is establishing the navigational links between and among the various pages. With automatically generated displays (such as diagnostic displays and status displays), the SCADA system will usually automatically create a directory list of all such pages. But graphical pages, with associations to specific (semi) automatically generated pages, usually require some user input to define these navigational linkages.

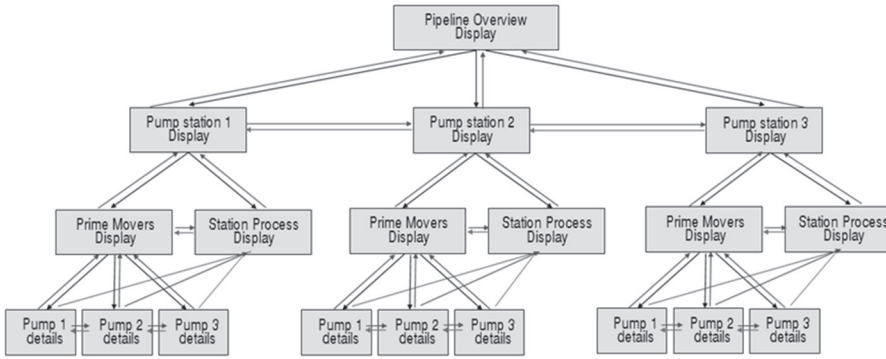


Fig. 5–10. Display hierarchy and inter-display navigation

Pan and zoom

In the electric utility industry (and to a lesser degree in the pipeline industry), a display navigation convention has become a standard. This is the use of displays that can be *panned* and *zoomed*. The electric utility industry was an early adopter of geographic information systems (GIS) and advanced display technology. All of the SCADA vendors serving that industry (at least for Energy Management Systems [EMS]) offer graphical display editors with the ability to define huge, highly detailed graphic displays that encompass a large geographic area and incorporate utility, geographic, topological, and physical infrastructure information. These graphical displays often require huge relational databases to hold all of the data associated with the display. The SCADA system vendor would purchase and integrate a GIS application and relational database application into their system and add functions to enable putting SCADA information into the GIS subsystem.

The graphical display presentation software is responsible for scaling and windowing the selected display view and creating an actual display image. A user can elect to enlarge or shrink the area displayed on their workstation, much like selecting a map with more or less detail. This is called zooming, and it establishes the magnification used for data presentation. Any display element that is too small at a given level of zoom is discarded and is not placed on the display. Zooming in to higher magnification will result in the reappearance of some formerly discarded display elements, although the area encompassed by the display window (the viewable area) will have shrunk correspondingly.

At a given level of magnification, only part of the entire display may be presented, and the rest is outside the display area. This is like looking through a window. You can only see the portion of the world framed in the window. Panning is the act of moving the window so that different portions of the overall display become visible, while some previously displayed areas are lost (like looking at the world through a different window). Figure 5–11 shows a map-based GIS display with compass-point

navigation, adjustable zoom level, and user-selectable layers. Incorporating GIS technology into the SCADA system's HMI usually adds significant initial effort because of the need to integrate field equipment, facilities, measurements, and other information into the GIS as one (or multiple) additional layer(s).

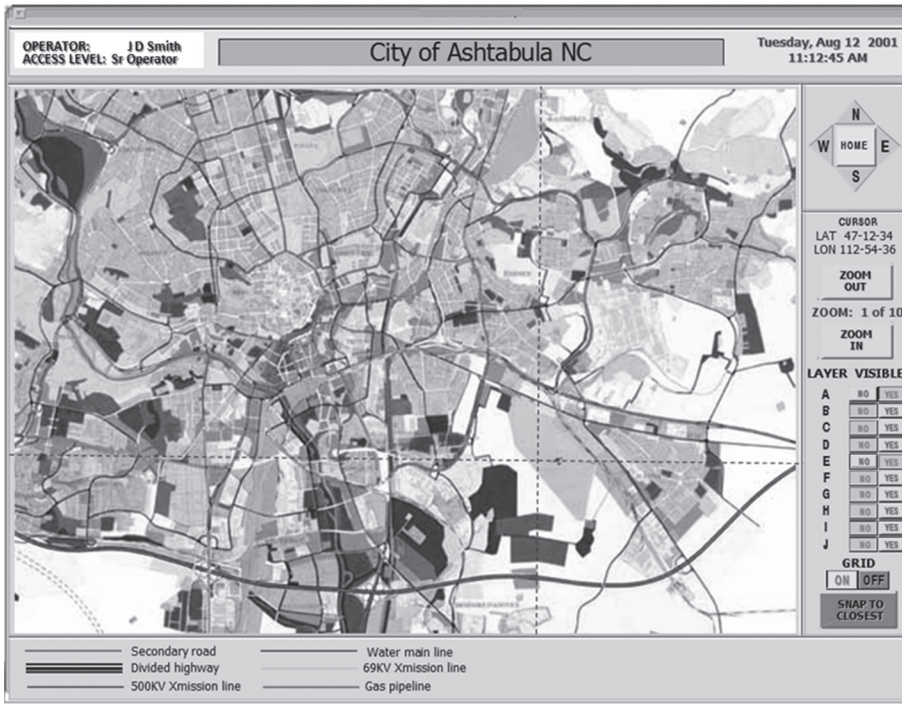


Fig. 5–11. GIS example SCADA display

Decluttering

Many electric and water utilities are multiservice utilities and have power, water, and even natural gas and cable television offerings. For this reason, it is often important and useful for them to incorporate all sorts of information, important to each of the service offerings, into their GIS displays. Thus, a view of a particular geographic area might be a composite of information about the physical facilities for all of these utilities, as well as pertinent geographic and infrastructure information, such as the location of roadways and waterways and utility poles. This much information can make for a very busy (cluttered) display and can distract the system operator from the critical points of interest. Thus, SCADA systems with GIS display capabilities usually offer the ability to add and remove categories (layers) of information. An operator interested in the electric power system might elect to remove the information about the water and sewer mains and the natural

gas pipelines, thus simplifying (decluttering) the display considerably. At a later time, someone from construction or engineering might want to see a composite of all of this information, for the purpose of laying out the path for a new electrical feeder. There is also an aspect of decluttering in the zoom feature, as objects will be removed from the display once they shrink below a specified size, as the zoom level decreases (the view becomes more distant).

Layering

The normal way to achieve decluttering is to allow the grouping of related information (e.g., all electrical infrastructure) into imaginary layers, with the ability to individually add or remove layers to display the desired view. The idea is similar to taking several clear plastic sheets, drawing separate information on each, and then laying them on top of each other to form a complete display. By removal of individual sheets (layers), the display can be simplified or specialized. In some instances, the act of removing display objects that get too small for practical display, due to the level of zoom, is also called decluttering,

Display navigation

GIS-based graphical displays are, by their nature and design, geographically based. Thus, navigation around the display is also in terms of geographic positioning. Most GIS displays offer some form of compass-point scheme with north/south/east/west selection icons (panning). There are also usually control icons for zoom manipulation and additional controls for layering and decluttering. In addition, it is typical to be able to click on objects and locations within the display and call up additional detailed information about those items or to cause a transfer to a different display that is specific to the selected item or location. With the advent of GPS technology, many water, gas, and electric utilities have begun storing GPS coordinates in their geographic/asset databases, so that an alarm received from a given location provides the display system with GPS coordinates and the display can automatically pan to the source of the alarm.

Alarms and indicators

Since SCADA systems tend to deal with large volumes of constantly changing data (2,000 - 70,000 or more tags, depending on the application), it would be infeasible, and far too time-consuming, for a human operator to constantly cycle through all of the data looking for problems. Therefore, it is important that the SCADA system provide at least basic validity and alarm checking on all of the data it is gathering (and even generating). Detecting alarm and abnormal conditions—and bringing these to the attention of the operators—is a primary function of SCADA systems. Part of the configuration work required as part of commissioning a SCADA system involves

defining the alarm thresholds to be used for each point. Some systems allow for a default set of alarm limits, and most also have built-in checks for invalid inputs. (For example, if an analog input is to be a 4 to 20 milliampere signal, then readings much outside of that range would indicate a potentially invalid and probably questionable input signal.) Figure 5–12 shows how as a given signal/measurement (or calculated) value traverses its expected (or possible) value range, it goes in and out of differing levels of alarm severity and indication. In most SCADA systems, unless some form of alarm inhibiting or disabling has been placed on an input, having its value transition across a pre-defined alarm or validity limit will cause some form of alarm indication and recording. This usually results in some form of visual indication on all displays containing that input, and a corresponding entry being made in a chronological log of all alarm events. Having the input's value return to a valid (normal) value is also usually accompanied by changes in the visual presentation and by another log entry. Some SCADA systems allow for re-alarming an input that has stayed in its current alarm state for longer than a specified time, so as to remind the operator of this continuing condition. A potential target of compromise for an attack on a SCADA system would be to disable the alarm check and annunciating functions (or to cause everything to go into an alarm state). This could potentially blind an operator to process upsets and unsafe conditions. With the alarming functions disabled, an operator could easily overlook a critical or dangerous condition developing in the field/process.

When most SCADA systems are initially powered up and initialized, because they have no prior data, they need to request an integrity poll of all of their RTUs, so that a complete set of real time data is developed. This process is usually time-consuming and generates a lot of spurious alarms unless the system has a means for suppressing them. Once this process is complete, most systems are generally stable and just continue the polling and alarm checking process. But during the initial start-up process, many SCADA systems are essentially unusable, and the operators blinded. If communications are lost to an RTU, many systems will mark all of the associated points in the real-time database in some manner that indicates that the data contained in that database, for those points, may not be trustworthy. (Some systems have an "offline" flag, some use a "questionable" flag, and some mark the data as "old.") In a less sophisticated system, this may also cause a flurry of alarms on all of the RTU's input points. When the RTU is restored to communications, the flags have to be removed, the RTU is commanded to perform an integrity poll, and again, there can be a flurry of alarms, depending on how things changed during the communication outage. Some systems provide a mechanism to allow alarming to be suspended/suppressed on operator-selected process areas (and preferably place a time limit on the suppression). Managing the way in which a SCADA system deals with alarms, and their impact on the operators, is an important consideration (and potential vulnerability). Remember that in most cases, the operator only knows that there is a problem because the SCADA system provides applicable alarms. If flooded with false alarms, the operators may be unable to determine an appropriate response or may take actions that turn out to be improper.

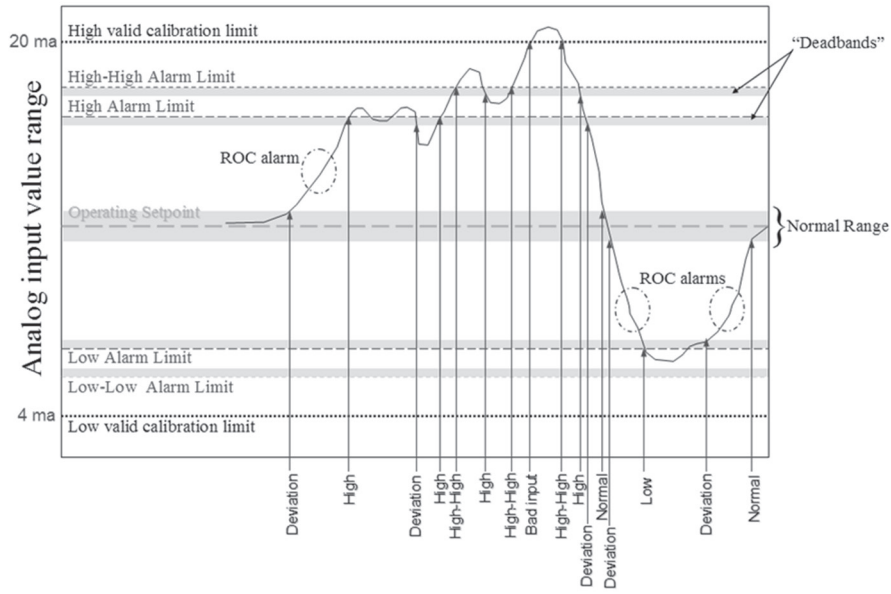


Fig. 5–12. Alarm limit checking on a typical analog input point

Analog inputs are generally given a validity check (to determine if the input can be trusted), followed by one or more sets of upper- and lower-allowable-value range checks (often called high- and low-level checks) and possibly a rate-of-change check and possibly an expected value (set point) deviation check. Most inputs to a SCADA system are assigned a set of fixed alarm limits, but some alarm limit checking may be based on the operational state of the process being monitored and may employ some form of mathematical model that is used to dynamically set and adjust the alarm limit values. The violation of alarm and validity limits is of differing levels of importance, based on the measurement in question. Since operators usually rely on alarm checking to alert them to potential problems, it is important that alarming functions be operating properly and that limits be properly set. If alarming functions were disabled, or limits changed to extreme values, dangerous conditions could go unnoticed by the operational personnel, resulting in damage, outages, and even threats to health and safety.

Another aspect of system initial configuration is the assignment of these alarm limits and of a severity level to each of the potential alarm conditions for each input. Unless properly configured, a SCADA system can generate a lot of nuisance alarms or fail to alarm critical conditions. When a system generates a lot of alarms, the operational staff tend to become insensitive to these alarms and either ignore them or disable the alarming functions, neither being a safe operational situation.

Contact and status inputs have similar alarm-checking provisions. Contact inputs can be checked for their current state (on/off) and for unexpected changes

of state. Contact input status changes may be considered as either alarms or events, depending on the circumstances. If the operator sends a command to start a pump and then receives a confirmation of the pump's status changing to RUN, then that status (contact input) change is not an alarm, but rather an event. On the other hand, if no such command was sent and the pump status were to change to RUN, that would be an alarm condition (un-commanded change of state). Operational personnel typically ignore events (which may not even be annunciated in any manner by the system), but most SCADA systems do log them. As with analog inputs, it is important that the alarm checking be properly performed on contact inputs as well. For some applications, in which precise (one-millisecond accuracy or more) time tagging is used on contact inputs, the determination—from a group of contacts which may be in different physical locations—of the first contact in the group that changed determines the alarm severity and type (called a *first out* alarm). In most SCADA systems, user-defined computed values (numeric and Boolean) are checked for alarm conditions each time they are recomputed, in much the same manner as physical inputs. When an input (or computed value) goes out of its normal operating range and into an alarm condition, the operational personnel need to be alerted; the method used is dependent on the severity of the alarm condition detected. Proper alarm checking on computed values can be as critical as that performed on physical inputs.

To ensure that alarms are actually noticed by operational personnel, most systems incorporate the concept of positive acknowledgment of alarms. When a point enters an alarm state, it is, by default, unacknowledged until an operator takes an action to acknowledge the alarm. This unacknowledged condition may be specifically indicated with a color or blinking condition on any display where the point is presented. Most SCADA systems differentiate between acknowledged (definitely seen by the operational personnel) and unacknowledged (ambiguous as to having been seen by the operational personnel) alarms. They may even be placed into different display pages to differentiate them even further. Figure 5–13 shows a typical alarm summary pop-up window with a couple of active, serious alarms (typically indicated in red), as well as an alarm that has returned to normal (typically indicated by green) without being acknowledged by the operator (indicated by its blinking condition). (Unfortunately, dear reader, it is difficult to show a color-coded or blinking image on the printed pages of a book.) Such displays usually present alarms in either chronological order or sorted by severity, and the display window alarm list will grow and shrink as alarms enter and exit. Many systems provide scrolling lists to accommodate large numbers of alarms. Many systems use blinking to indicate unacknowledged alarms, and this persists even if the point returns to a normal value, until acknowledged by an operator.

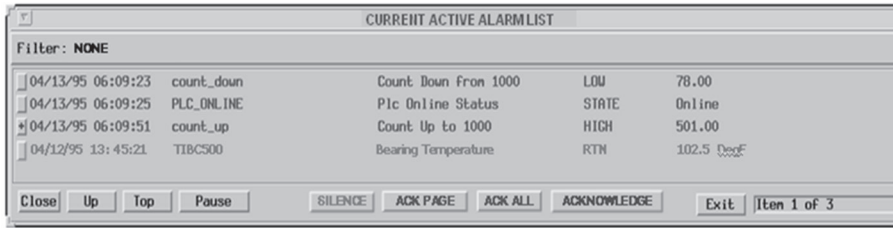


Fig. 5–13. Typical current-alarm summary display window

Alarm filtering

In the event of certain types of operational or process-upset conditions, there may be a flood of alarms from the field, and this can distract operational personnel. This could be caused by an actual problem or by such activities as shutting down a process area or just turning off an RTU for maintenance. In a large SCADA system application that supervises an extensive geographically distributed process, different operational personnel might be responsible for different subsets of the overall process. In those cases, it is important that a given operator not be distracted by alarms and notifications related to a subset for which he or she has no responsibility. (For example, if Joe is responsible for the northern district, he doesn't want to be distracted by alarms on his console coming in from the southern district, which is the supervisory responsibility of Bob.)

SCADA systems have incorporated mechanisms for reducing and focusing the information presented to operational personnel. *Alarm filtering* is one such mechanism. With most systems, the operational personnel will have the ability to designate the logical or physical process areas from which they do, or do not, wish to receive alarms. They may also be able to define categories or severity levels of alarms that they wish to be excluded from their displays. When a field site is disrupted by on-site work, it is desirable to prevent spurious alarms from being generated at the SCADA system every time an on-site worker turns the power to a piece of equipment or an RTU on or off. It is usually possible for a SCADA operator to select predefined filter options from a list or to define customized filter options.

Of course, it is important that operational personnel be aware that alarms are being filtered and that alarm filters require some periodic reapplication, so they aren't forgotten and left in effect by accident. A possible cybersecurity vulnerability of SCADA systems would be having the alarm-monitoring/presentation settings modified to filter out all alarms, since in many cases, operational personnel depend primarily on alarms to direct their attention to problem areas. An equally dangerous alternative would be causing the alarm-detection software to flood the operator and operational displays with false alarms. The overall issue of display alteration or information manipulation is a general cybersecurity concern, since

operational displays provide the primary mechanism through which operational personnel identify and correct emerging process problems.

Alarm annunciation

When new alarm conditions are detected, it is incumbent on the SCADA system to bring these to the attention of operational personnel. One of the primary means for accomplishing this is to place the new alarm information into the active/current-alarm summary list and display. Another commonly employed mechanism is to generate some form of audible signal that will attract the attention of operational personnel. This may be a set of unique sounds played through the speakers of the operational workstation(s) or an audible alarm signal generated externally. To stop (silence) this audible signal, the operational personnel would normally be required to acknowledge the new alarms and possibly operate a separate control to silence the annunciation.

For particularly serious alarm conditions, some SCADA systems might also extend the range of personnel to be notified (outside the immediate control room), by using mechanisms such as personal pagers, cell phone text messaging, and email to send automated alarm messages to relevant senior technical, management, and supervisory personnel. The danger in utilizing email (and some forms of cell phone) notification is that there must be IP connectivity—from the SCADA system to a corporate mail server, and maybe through that server to the Internet itself, depending on the recipients—to make email delivery possible. As we will discuss in a later chapter, in IP networking, having other IP-based systems between you and the Internet doesn't mean you are protected from attacks coming from the Internet. Similarly, in order to make use of pager systems or cell phone text messaging, a SCADA system would typically need a permanently connected telephone circuit (or a cellular gateway/modem) so that it can call the pager service or send a text message. This also opens up a potential vulnerability and attack pathway for cyber attackers to exploit, using an approach called *war dialing*.

Alarm history file

At any given point in time, there will typically be some subset of the entire set of inputs and calculated values that are currently *in alarm* (outside their normal, acceptable operating range). This list will vary over time, as inputs return to a normal (non-alarm) condition and others enter an alarm condition. An alarm summary display (fig. 5–13) will usually be provided, to list the signals and equipment currently in an alarm state. (Most systems will purge from this list points that return to normal, unless they have not been acknowledged by the operational staff.)

It is also useful to keep track of all the alarm comings and goings, as well as the supervisory actions of operational personnel. Most SCADA systems have some form of historical logging function, whereby alarm transitions, events, alarm

acknowledgments, operator actions, and system messages are placed into one or more chronologically ordered tables so that this information can be reviewed, to look for situations that indicate a process problem, an operational problem, a training problem, or a system problem. These logs are not associated with, or for the same purpose as, operating system logs and device Syslog messages (and they are not usually fed to a SIEM if one is being employed as part of overall cybersecurity). A specific and useful log maintained by most SCADA systems is an operator action (or tracking) log wherein a record is kept of the parameter adjustments, alarm acknowledgements, control commands, and other actions taken by each operator. This log is especially helpful when keeping track of different operational shifts, evaluating the performance of operator trainees, and keeping an audit trail of remote personnel with operator control authority. In the event of a problem in the field or an accident, this log could provide evidence in a legal proceeding. Preserving this log from destruction and alteration is an important cybersecurity consideration. There was a municipal sewage treatment plant that had a spill into an adjacent waterway due to a waste tank being allowed to overflow. The operator on duty (who was in a dispute with the municipality over a denied promotion at the time) claimed that the SCADA system malfunctioned and never provided any alarm about the tank level. A review of the operator log showed that several alarms were in fact generated as the tank level rose and that each alarm had been acknowledged and silenced in turn by the operator in question. SCADA systems usually provide a means for displaying, sorting, filtering, and printing these tables. Deletion or alteration of such tables should require administrative level access.

The term *log* is usually used to describe the printed results of such operations. Most SCADA systems provide a range of predefined logs and even some level of configurability in formatting the resulting printed documents. In earlier ages of SCADA technology, a system might come configured with several printers, each dedicated to a particular class of log—with boxes of form-feed, fanfold paper providing a continuous, printed, chronological audit trail of alarms, events, operational actions, and system messages. Storing this information in relational database tables, rather that printing it onto reams and reams of paper, has undoubtedly saved a great number of trees and made the information much easier to sort, organize, search, and utilize. Nevertheless, some SCADA system owners still prefer the use of printed logs, because they can be harder to manipulate and falsify (or delete—but not destroy) than data stored in a computer file or relational database table.

Alarm-state visual indication

In every SCADA system, color and blinking (as well as other visual means) are generally used in the operational displays to indicate current alarm status. If a measurement is present on an operational display and is within its normal operating range, the system may be configured to display the value of that measurement in green (although this varies a bit by industry). Red, yellow, and other colors are often

used to indicate parameters that have entered an alarm, or questionable, condition or which are disabled or possibly set for manual override. In earlier SCADA systems, for which color-enabled video display hardware would not have been readily available (in the 1970s), alarm conditions were indicated by using reverse video and by adding code letters adjacent to the displayed value.

Today, in addition to color and blinking, special icons or symbols may be appended to a displayed value to indicate specific conditions. As an example, if an input point has been taken off scan, or is having its value manually overridden by the operator, or it has been tagged and blocked from the operator, then those conditions will typically be indicated as well. Every display in which a given parameter or point is presented should incorporate the same set of color/symbol/letter codes for that parameter or point. Since many conditions can be present concurrently, it is not unusual to see combinations of color, icons, and code letters being employed. Figure 5–14 shows examples of point-presentation color-coding and labeling variations. One reason for using symbols and text labels in addition to colors is that some people are color blind and may unfortunately be unable to tell the difference between certain color combinations.

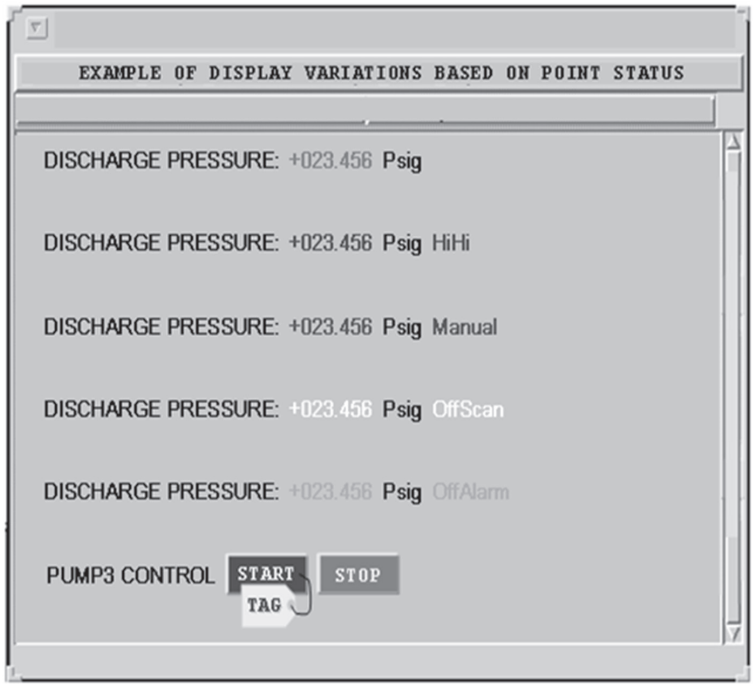


Fig. 5–14. Using symbols or code letters to indicate measurement conditions

For control (contact and analog) output points, it is common to show a tagged indicator next to the control point on all displays where that control is presented and available for control action initiation. *Tagging* is the process of disabling a control-output from any form of supervisory control (operator- or application-initiated) for the purpose of equipment and/or personnel safety. The term derives from the practice of placing actual, physical paper tags onto control handles on equipment, so that people would see the tags and avoid operating or activating the equipment. In theory, tagged control outputs cannot be operated through the SCADA system until all user-assigned tags are removed. Tagging is common in the electric utility industry and has become more common in other industries over the past two decades, as SCADA vendors introduced their technologies into other market segments.

Enforcement of tagging can be implemented at various levels on a SCADA system's software. The most robust implementations use information written directly into the configuration database of the RTU (or PLC) that contains a tagged control point, so that checking for a tagged condition happens right at the level on which actual manipulation of those physical outputs takes place. SCADA systems usually allow multiple tags to be assigned to any control point, by different personnel or functional groups (e.g. maintenance, operations, electricians), and *all* such tags must be removed to re-enable supervisory control of any given control output. SCADA systems that implement tag enforcement at the host level (which is a commonly used design) or the operator permission–checking level are potentially vulnerable to tag bypassing. An attacker who gains access to the communication channel (or WAN) is not prevented from sending control commands to the RTU and having them acted upon, because the RTU is unaware of tagging if tags only exist in the SCADA host. Figure 5–15 shows tagging activity being initiated, via the operator's display, on a circuit breaker in a substation. The ability to violate tagging creates the distinct possibility of causing damage, serious injury, or even death to on-site personnel. (Unfortunately, that is a very newsworthy and frightening event, the kind of thing a terrorist would like to cause.) Many organizations don't fully trust the SCADA system, even if it properly supports tagging, and still require manual disabling of controls in the field, possibly by switching off control loop power at the RTU prior to performing potentially dangerous work activities.

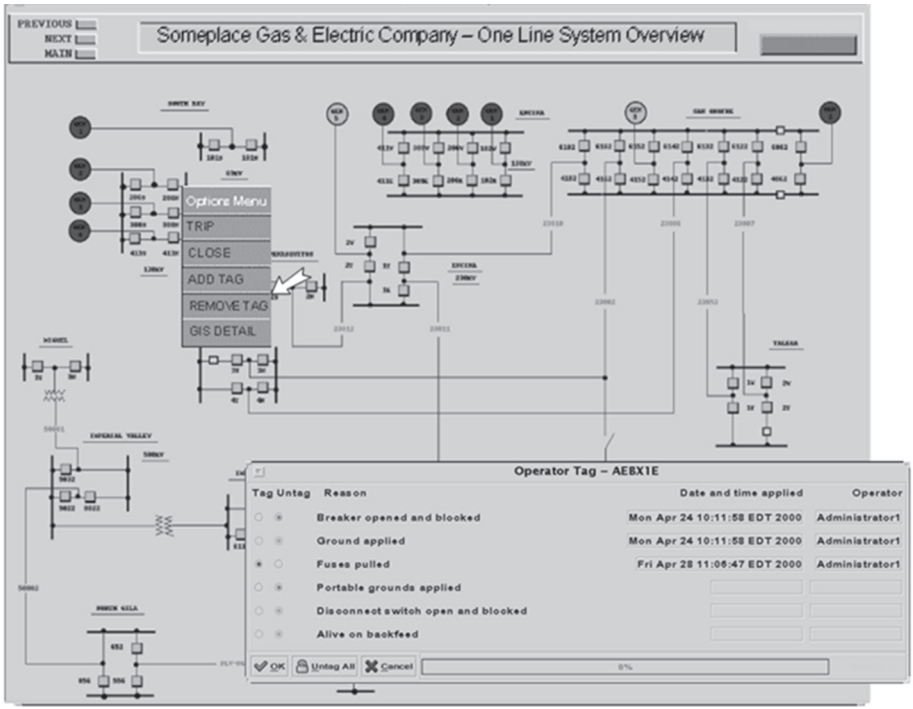


Fig. 5–15. Control-point tagging display

Historical Trending

SCADA systems collect a constant stream of updating real-time data from the RTUs in the field and update (overwrite) their real-time database with these new values. Often, however, just as important as knowing the current value of a measurement is knowing the (recent to distant) past value changes (history) of many of those same measurements. Knowing that a tank level is currently at 15.56 feet doesn't provide the same understanding of what is happening as being able to see that this level has dropped (or risen) to that point over the past few minutes or past several hours. All SCADA systems incorporate some level of historical data recording, whereby the operational personnel have the capability of reviewing the past history of a selected set of key measurements (or calculated values) for some predefined time span extending into the past. Prior to SCADA systems, one of the primary mechanisms for monitoring a remote measurement was to use that measurement value to move a pen up and down across a strip or disk of moving paper. Figure 5–16 shows a typical electromechanical, multichannel, strip-chart recorder from the 1980s. The historical trending functions and displays of most SCADA systems essentially mimic the actions of those obsolete (but still used)

electromechanical devices. The ability to dump recorded data off the SCADA system and onto recordable removable media (e.g., a DVD platter) for long-term archive storage was also critical for eventually eliminating the electromechanical recorders because the paper charts actually provided a legal record of sorts, which the DVDs could replace.

Fig. 5–16. Mechanical strip-chart pen recorder

Strip-chart and circular-chart recorders were invented for pre-computer data recording purposes and have remained a staple of the process-monitoring and control industry even up to the present day. The historical trending and display functions of a SCADA system are an electronic reproduction of that older technology. Today, historical trending (historian) packages are also available from third-party vendors, either as application software to run on the SCADA system or as a separate server/system that can interact and exchange data with the SCADA system. In prior decades, SCADA system vendors generally had to develop their own proprietary versions of data historians. A major aid in doing this came in the mid-1990s with the evolution of commercial relational database packages that had adequate performance capabilities—and the massive increase in the storage capacity of disk drives.

Historical trending involves four aspects: initial configuration of the data collection and display functions; actual collection of the specified data values; storage and management of the collected data; and finally, the presentation of the recorded data to operational personnel. Computers have dramatically increased

in computing power and storage capacity over the past few decades. But, in spite of those advances, computers still have practical and physical limitations. It may not be physically possible or necessary to attempt to collect and continuously store every measurement, status input, and computed value within the SCADA system. No matter how powerful your computer, you can bring it to a halt by placing upon it too great a processing and data manipulation burden.

Similarly, it is not possible to equip a computer with an infinite amount of mass storage (although these days, the amount possible, using NAS technology, can be quite massive). For those reasons, most SCADA system historical trending packages require the user to pick a subset of the available database points for collection and then allocates a given amount of storage for trending purposes. These selected points will be the only measurements assigned to historical trending. The time span over which the trending can be maintained will depend on the number of points being trended, the rate of trending, and the amount of storage allocated. All field measurements are not equal in their importance or in the rate at which they can change. A bearing temperature could be very important and might change rapidly. The water level in a lake might also be important, but it is unlikely to change very rapidly under any reasonable conditions. Therefore, a user may wish to store samples of these different measurements at a different collection periodicity.

Historical trending packages often manage the storage and collection process by offering prespecified *trend groups.* For example, a trending package might allow data to be written to historical storage from the real-time database at a 1-, 5-, 15-, or 30 minute collection rate (i.e., one sample stored to the trend files every 1, 5, 15, or 30 minutes). Obviously, over any fixed time period, the data collected once per minute will be of a much greater quantity (60 times as much) than that collected once per hour, if the number of trended parameters is the same for each collection rate. Such a trending package might place maximum limits on the number of parameters that could be assigned to each collection rate, thus predefining the maximum amount of disk space that could be used by the historical trending function. (For example, for Group 1 [data collection every 1 minute], up to 500 points may be assigned; for Group 2 [data collection every 5 minutes], up to 2,500 points may be assigned; and for Group 3 [data collection every 15 minutes], up to 6,000 points can be assigned.)

Once the predefined maximum amount of data has been collected, there is, in theory, no further storage space available, so either historical trending must cease, or additional room must be made available. In many systems, the storage that is pre-allocated is treated as if it were logically structured into a *circular buffer*: when the end of the buffer is reached, the subsequent data are stored at the beginning of the buffer, overwriting the oldest previously recorded data values. Using storage in this manner prevents the filling of all available storage space (and shutting down of the system), while ensuring that there is a specified guaranteed amount of prior history (going from now, back through a given amount of time) available for any given measurement assigned to historical data collection.

Another way in which trending packages optimize the required storage space is by reducing data quantities through statistical data manipulation (e.g. taking all of the data samples collected every minute for a 30 minute interval (30 values) and reducing them to just three values: the average, maximum, and minimum values for that 30 minute interval). Figure 5–17 shows a variety of data archiving, storage, and reduction methods for long-term historical recording purposes. The purpose of collecting and archiving data is to provide a historical perspective on the actions of critical measurements and values. Historical trending applications provide a limited look back in time for the values being archived. How much you can trend and hold online will depend on your SCADA system design and architecture. If your SCADA system consists of one desktop PC, then you will not have the historical trending capacity of a SCADA system that allocates a LAN-connected storage server, with huge hard drive capacity, specifically for the historical trending function.

Conventional SCADA systems with low-speed serial communications to the RTUs had limits on how fast data updates could be collected from the RTUs, and so historical trending sampling speeds had that same constraint (if a data value can only be fetched from an RTU every 60 seconds, it makes no sense to save it as a trend sample every 10 seconds in the host). But, if a SCADA system has high-speed WAN connectivity to the field sites and RTUs capable of network support, then it is viable to consider speeding up both RTU polling and then historical trending sampling rates can also be much faster. This is not to say that you need to collect/sample more frequently, only that it might be possible if that were desirable. Of course, more frequent trending sampling means more disk space will be needed to hold the data.

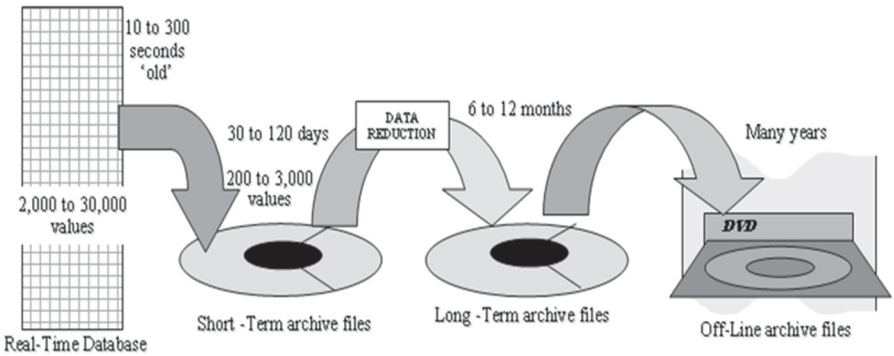


Fig. 5–17. Data storage hierarchy for historical trending

It may be important to have 1 minute data samples on a given point *x* for the most recent 24 hours. Beyond that time span, it may not be as critical to collect data that often, so sampling and collection of *x* values could be reduced to once

per hour. One can achieve this by placing point *x* into both a 1 minute collection group and a 60 minute collection group. This approach leads to the problem of storing more data than is necessary (although when storage is plentiful, this may not matter).

When trend storage is extended over a long time interval (many months, multiple years), and a slow sampling rate (1 minute, 5 minutes, etc.) is selected to reduce the storage required, there is always the potential problem of missing something important during the intervening time between sample captures. Some SCADA systems address this problem by computing, for example, the hourly value (rather than fetching the current value at the hour demarcation), using the minute data collected in the prior hour. This approach facilitates deriving a more representative value, rather than merely saving the instantaneous reading at each hourly point. The average/mean, maximum value, or minimum value (or all of them) can be computed from the prior hour's data and used as the hourly recorded value(s).

Many SCADA systems offer an alternative to merely overwriting (and losing) the oldest historical data as additional newer data is stored. In these systems, prior to overwriting (and losing) the oldest data values, those values are copied onto some form of removable storage media. In older SCADA systems, this might have been a magnetic tape cartridge, but today it's more likely a digital versatile disc (DVD) (fig. 5–17) or a large-capacity USB flash drive or solid-state hard drive. The data preserved in this manner are no longer immediately available for display on the SCADA system, but if needed at some point in time, the data can be copied (temporarily) back onto the system, or the removable media inserted into the appropriate removable media drive and made available for display. Data that are not immediately available but must be brought back into a system via some manual intervention (and possibly a data transfer process) are usually called *offline* storage. Data that reside on the hard drive(s) and can be immediately accessed by application programs are usually called *online* storage. Removable, directly readable media, such as DVDs and USB flash drives, have blurred this distinction a bit. The data contained on them is not immediately available, but there is no requirement to copy the data onto the SCADA system in order to access this data. There is no practical limit on how much data you can retain offline as long as you don't run out of removable media and shelf space. But how much data you can retain online will depend on available bulk storage. It is not uncommon, with the hard-drive capacities of today, to have some level of online historical data that spans the last 12 to 24 months of operation. A single analog value sampled for historical trending on a 1 minute sample rate with its values retained for 24 months would require about four [4] megabytes of hard drive storage space. With multi-terabyte-capacity hard drives available today, this means you could have thousands of measurements stored at that sample rate for that time span.

In some industries, it is of regulatory and legal value to maintain historical data records with no limit on the time span being preserved. Using offline storage as a

permanent means for data archiving addresses this need, without placing undue demands on the production system (or exposing this data to cyber tampering or destruction). In some cases, the historical trend data may be of financial importance (e.g., used for documenting that contract pressures or flow rates were maintained or for totalizing computations), and therefore its loss or alteration could have a potential financial impact.

Historical trending displays

Once historical data are collected by a SCADA system, they are available to the operational personnel, usually in the form of trend (time plot) displays. A trend display usually provides some description of the data being presented, as well as some form of scale or grid to determine the values being plotted. Data are presented in the form of an *X-Y* plot of values (on the *Y* axis) against time (on the *X* axis). Most historical trending packages allow multiple signals to be co-plotted on the same display grid for comparison purposes. Most historical trending packages allow the user to rescale the *X* and *Y* axes to obtain the desired resolution and data clarity. Most also permit the scrolling of data forward and backward in time, within the constraints of the available data. (No, you can't scroll into the future to see what is going to happen.) Figure 5–18 shows a multi-pane, multi-trace historical trending display of various measurements associated with transformer performance.

There are a wide range of more advanced functions that can be found in various historical trending packages, and it is not the goal of this book to enumerate all of these features. A historical trending package deals with fixed data, recorded at some time in the past, up to, at most, the present moment. For that reason, such trend displays generally are static. (Once the requested data has been presented, the display has no reason to change or update.) Most packages also allow for some form of *quality code* to be attached to the stored data. This is used to indicate any issues concerning the validity or integrity of the stored data values. For example, if stored values were from physical inputs that were being manually overridden, a code might be used to indicate this state. If no data were recorded over a time interval, the display may leave a corresponding gap in the trend plot. The specific features, formats, and functions will depend upon the SCADA system vendor, unless they use a third-party commercial historian package.

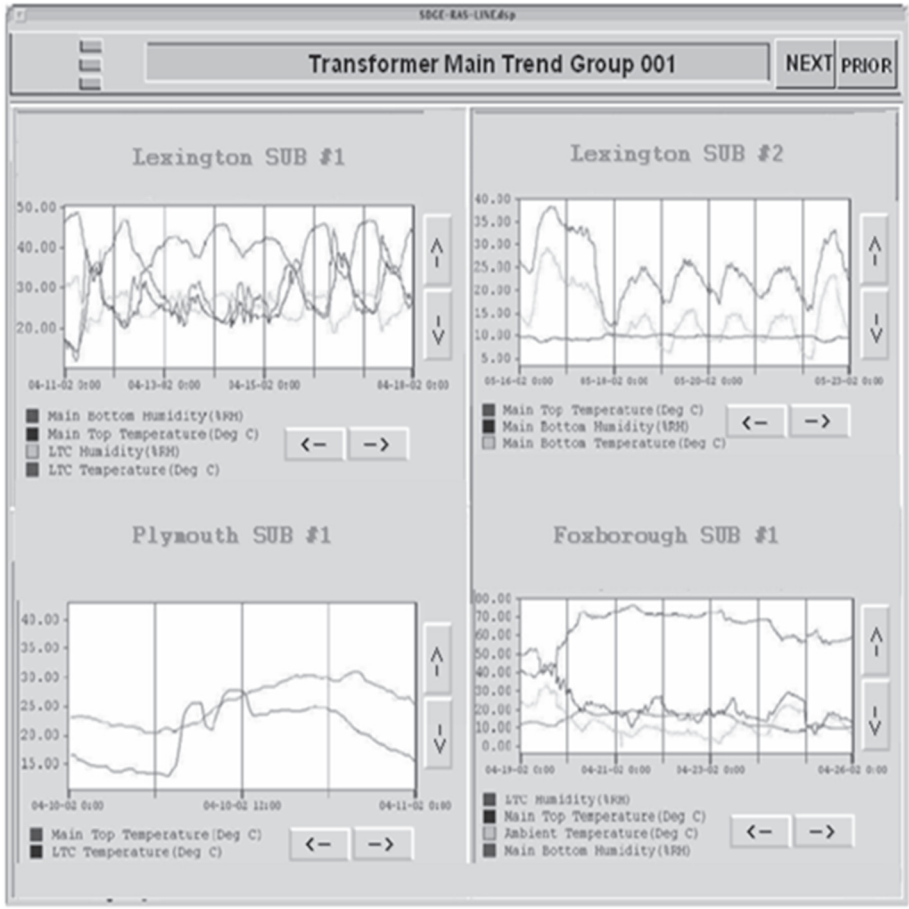


Fig. 5–18. Example historical trending display

Real-time trending

There are also occasions when it is useful to continuously monitor a measurement (or several measurements) for a period of time—watching what the values have been recently, as well as the current value(s). With real-time trending, the plotting of data values typically begins when you call up the display and then the plot updates as the next values are received. When the plot has filled the width of the display window, it will typically shift thereafter, losing the oldest value and adding the newest value. So, unlike historical trends, real-time trends are active displays that constantly update. Such trending can be supplied either as an extension of the historical trending package or as a separate capability. The term *real time* must be interpreted in light of the processes being monitored and the communication capabilities of the SCADA system. Because of the generally

slow-speed communications between the SCADA host and the field-based RTUs in some industries, fetching new data values is usually what limits the ability to provide real-time trending. In the water/wastewater industry, a real-time trend may update only every few minutes, whereas in the electric utility industry, trend updates may be possible every few seconds—simply because of the differences in the communication architectures and technology. As was already mentioned, with IP networking to the field, greater data collection rates may be possible, and this would directly impact real-time trending.

An important consideration when configuring a real-time trending package is to avoid attempting to collect and store data at a rate that is faster than the actual values are being updated by polling of the RTUs. The process of updating the real-time database is typically asynchronous of (disconnected from) the process of writing database values to historical files. An RTU could scan and update a particular measurement every 10 seconds and be polled by the SCADA host for that value every 30 seconds (so the data could be up to 30 seconds old in the real-time database of the SCADA system), and yet the operator has it on a trending display that updates at a 10-second rate. An operator might be fooled into thinking that a process, or particular measurement, was stable because a real-time trend plot of values was holding steady. However, in reality, communications to the particular RTU might have actually been lost. In most SCADA systems, the historical and real-time trending packages normally fetch their data from the real-time database that is refreshed by polling RTUs. So, it is important that the status of data in the real-time database be included and updated, and not just the value.

As with any operational display, it is important to somehow alert a system user that the data being viewed could possibly be invalid or questionable or is just not available. In SCADA control rooms, real-time trending displays, just like alarm summary displays, are a critical component of the overall operational data presentation.

In some Microsoft Windows-based data historian systems, OPC classic may be used to link a the trending server to its data sources (including the SCADA system). A third-party commercial historian/trending package, external to the SCADA system, might receive data updates from one or more sources, over a LAN connection. Other third-party historian packages may employ Modbus-over-IP or other IP-based communications to receive their data updates. A potential cybersecurity vulnerability in SCADA systems that use this type of external historian package/server is an attack that disrupts such communications or that sends falsified data to the historian. In these instances, the historian is a separate computer, in which case if it is not suitably and adequately protected, it could also be subject to a direct cyberattack and if compromised, could then be used as a platform to attack the SCADA system.

Logs and Reports

Aside from the use of SCADA systems to provide real-time monitoring and supervisory control, a major objective for most systems is to utilize the collected data to automatically generate (printed) reports and/or logs—for corporate purposes, regulators, government agencies, and other organizations that require regular reporting. In the water/wastewater market, it is not uncommon for daily, weekly, quarterly, and annual reports to be required by the appropriate regulatory agencies. All businesses need regular reporting on operating costs, inventory, actual production, asset productivity, and asset utilization. Since a SCADA system collects and processes data from many sources, it is a logical place to generate the reports that require these data.

It is a good idea to differentiate between logs and reports. Logs are a chronological accumulation of associated information, sorted by category—generated by the occurrence of conditions that are defined as abnormal or worthy of recording. They are, in effect, an audit trail of monitored conditions. If, during a given time interval (e.g., a day or a week), no occurrences of monitored conditions are detected, then there is nothing in the log. Reports, by contrast, are a predefined set of data and computations based on those data, that are to be put into a predefined format (and probably printed), either on a scheduled basis or on an event-triggered basis. When a report is generated, it should not be blank unless problems prevented the collection of the specified data. An empty log may be a good thing; an empty report (one with no data) is not. Figure 5–19 shows a demand log for a given operating area (the user-selected filter criteria) listing control events recorded over a user-specified time interval of 24 hours. The logs created and presented/printed by the SCADA system are different, and usually separate, from those that are generated by the operating system of the computer(s) on which the SCADA software is running. Some SCADA packages actually write their logs into the Application log file of their Windows O.S. so that standard Windows utilities such as the Event Viewer can be used to search and sort those logs. But if many applications are also writing to that file, the SCADA entries could be harder to specifically identify.

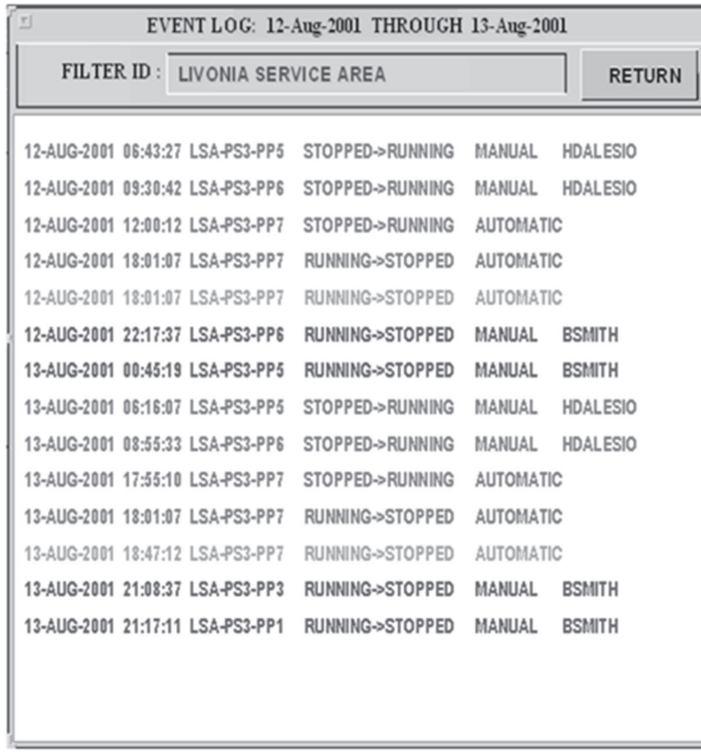


Fig. 5–19. A typical SCADA system event log query

Calculated values

Although most reports involve collecting and presenting a predefined set of data, most useful reports also require that some level of calculations be performed on the data. An hourly list of water quantities delivered to custody points will typically include a total for each custody point and a total across all delivery points for the day. Addition is a simple mathematical function, but some mechanism is needed to perform the calculation. In some older SCADA systems, all such calculations needed to be created using a separate user-defined calculated point facility. Any computed values needed for report generation had to be defined in the form of computations that generated pseudo-analog or binary database points, and these could then be referenced in any subsequent reports (which added significantly to the overall size of the point database of the system). Today, most reporting packages have integral calculation capabilities, much like those found in popular spreadsheet programs. Some SCADA packages actually provide a means for delivering data to a commercial spreadsheet application, such as Microsoft Excel™, so that the full range of spreadsheet computations, logic, and formatting can be used in report generation.

Statistical calculations

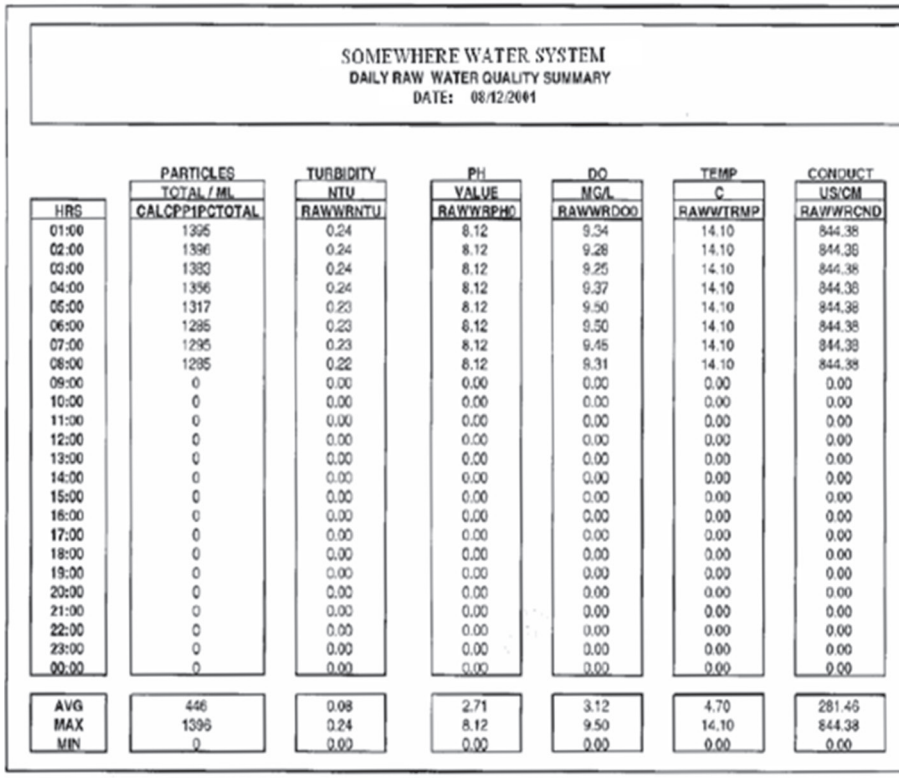
Frequently, the mathematical processing of data involves statistical calculations, such as computing the mean, the maximum, or the standard deviation for a set of values. This set of values may actually be extracted from the historical trend files for a user-specified single measurement, over some selected time interval (e.g., the prior day or week). In these cases, the reporting package needs to be able to interact with the historical trending package to fetch the necessary data points. Again, in some systems, this can be handled in the reporting package, but in others, such computations need to be performed with separate capabilities and made available to the reporting package. If very complex calculations are required in order to generate a value for a report—particularly a calculation that involves multi-pass algorithmic logic (a mathematical algorithm)—then once again, this may be done in separate software and a value made available to the report package.

Spreadsheet report generators

With the migration of SCADA systems to the Microsoft Windows and Linux operating system platforms, it has become possible to make use of commercial spreadsheet software packages for report generation. These packages all have several forms of data import mechanism and allow vendor-written add-ins that can be used to connect them to SCADA system data sources (or by collecting and placing such data into a file where the spreadsheet can use its import capability). The computational and output-formatting capabilities of such packages are prodigious and very flexible. From a cybersecurity perspective, it should be noted that Excel spreadsheets can have VBscripts embedded in them and this scripting will be executed when the spreadsheet is displayed by Excel. If an adversary were able to substitute a malicious Excel file for a legitimate report file, then when the report was next run, the script would execute. Microsoft has given VBscripting almost unlimited access to system objects, and they can be used do great harm to a Windows-based system.

Almost any reasonable report can be defined using the capabilities of modern spreadsheet packages. Most current SCADA systems make use of (*integrate*) these commercial spreadsheet packages. Figure 5–20 shows a daily water production report (taken partway through the day) giving hourly breakdowns of volumes and water-quality information. Most commercial relational database packages support an ODBC (Open Database Connectivity) interface, as do most commercial spreadsheet packages. SCADA systems that maintain all of their data, even the real-time data, in relational database tables make it easy to select and fetch the desired data into a report spreadsheet. A final requirement for reporting is to have a means for triggering reports at pre-specified times and dates.

Both Windows and Linux support task scheduling, Windows with the Task Scheduler tool (fig. 5–21) and Linux with the Crontab function. Both allow you



SOMEWHERE WATER SYSTEM
DAILY RAW WATER QUALITY SUMMARY
DATE: 08/12/2001

HRS	PARTICLES TOTAL / ML CALCPP1PCTOTAL	TURBIDITY NTU RAWWRNTU	PH VALUE RAWWRPH0	DO MG/L RAWWRDO0	TEMP C RAWWTRMP	CONDUCT US/CM RAWWRCND
01:00	1395	0.24	8.12	9.34	14.10	844.38
02:00	1396	0.24	8.12	9.28	14.10	844.38
03:00	1393	0.24	8.12	9.25	14.10	844.38
04:00	1356	0.24	8.12	9.37	14.10	844.38
05:00	1317	0.23	8.12	9.50	14.10	844.38
06:00	1285	0.23	8.12	9.50	14.10	844.38
07:00	1295	0.23	8.12	9.45	14.10	844.38
08:00	1285	0.22	8.12	9.31	14.10	844.38
09:00	0	0.00	0.00	0.00	0.00	0.00
10:00	0	0.00	0.00	0.00	0.00	0.00
11:00	0	0.00	0.00	0.00	0.00	0.00
12:00	0	0.00	0.00	0.00	0.00	0.00
13:00	0	0.00	0.00	0.00	0.00	0.00
14:00	0	0.00	0.00	0.00	0.00	0.00
15:00	0	0.00	0.00	0.00	0.00	0.00
16:00	0	0.00	0.00	0.00	0.00	0.00
17:00	0	0.00	0.00	0.00	0.00	0.00
18:00	0	0.00	0.00	0.00	0.00	0.00
19:00	0	0.00	0.00	0.00	0.00	0.00
20:00	0	0.00	0.00	0.00	0.00	0.00
21:00	0	0.00	0.00	0.00	0.00	0.00
22:00	0	0.00	0.00	0.00	0.00	0.00
23:00	0	0.00	0.00	0.00	0.00	0.00
00:00	0	0.00	0.00	0.00	0.00	0.00
AVG	446	0.08	2.71	3.12	4.70	281.46
MAX	1396	0.24	8.12	9.50	14.10	844.38
MIN	0	0.00	0.00	0.00	0.00	0.00

Fig. 5–20. Example of a spreadsheet report for a water utility

to schedule tasks, such as launching a spreadsheet application, based on date and time. Both give you flexibility in how you define the date, using day of the month and day of the week combinations.

For reports that need to be triggered by events, the SCADA system software would need to provide a triggering mechanism that is user definable/selectable since the computer operating system knows nothing about the SCADA functions or SCADA system-identified events.

The vast majority of operator interface functions and capabilities in a SCADA system are defined and configured by a range of files and databases. If these are corrupted or just deleted, then the system would need to be shut down and restored from backups. In a redundant configuration, that would mean switching to the backup system. But an attacker would probably be aware of your having a redundant configuration, and a well-designed cyberattack would ensure that both systems were corrupted. This is one reason why we will discuss the need to establish LAN segmentation and protections to ensure that this won't happen. This architectural strategy will be discussed in a subsequent chapter.

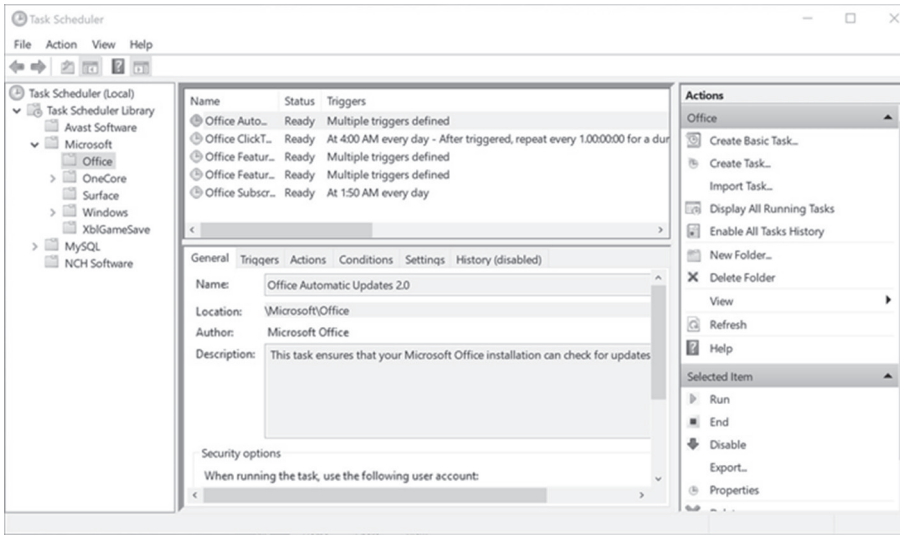


Fig. 5–21. Microsoft Windows Task Scheduler Utility

6

Conventional information technology

SCADA systems have migrated to commercial hardware and operating system and networking platforms, and they incorporate many well-known, commercially available components (e.g., relational databases and Web browsers and servers). In effect, a SCADA system today is architecturally identical to any other IT system, except for the specialized application programs that make it a SCADA system (versus, for example, an accounting system) plus the need for 100 percent availability. Thus, it is not surprising that many, if not most, of the techniques and technologies employed in securing IT systems have direct applicability to modern SCADA systems. That is not to say that all of them are applicable. Certain procedural approaches that are considered normal in the IT world would be unacceptable in the SCADA world (e.g., applying all patches issued by any software vendor without testing them on a development system first or just rebooting a system whenever it seems to be operating strangely). In order to properly protect a modern SCADA system from cyberattack, we need to have a basic understanding of how IP-based networking (and Ethernet LANs) operate, of typical network components, and of where there are exploitable features and functions. In this chapter, we will build a basic understanding of these technologies.

Many organizations that use SCADA systems also usually have formal corporate IT support departments. In the past, there has been a lack of trust and cooperation between the personnel who run and maintain the SCADA systems (sometimes referred to as OT personnel) and the corporate IT personnel. There are historical reasons for this situation. Large and complex SCADA systems can be difficult to configure, commission, and put into operation, and once these systems are operational, most SCADA system users are loath to allow any changes, unless made specifically to correct a known and intolerable bug or malfunction. The methodologies of an IT group may not match well with this mentality. Classically, IT groups immediately deploy and install all patches and new software releases, to keep all systems up to date. They also like to mandate configuration and application *baselines* and standards that may not be in keeping with the software supplied/required by a SCADA system vendor. For these and many other reasons, interdepartmental clashes result, much to the detriment of the overall organization.

Today, with SCADA systems utilizing much of the same technology that is employed in IT systems—and needing to be interconnected with other corporate

and external systems via LAN, WAN, and Internet technology—it is important that IT and the operations group (the typical users of the SCADA systems—also often referred to as 'OT') cooperate in making SCADA systems secure from cyberthreats. To foster such cooperation, SCADA system personnel should understand the basics of cybersecurity as practiced by most IT organizations (and IT personnel should be encouraged to listen to the issues posed by the SCADA personnel).

Availability, Integrity, and Confidentiality

When IT security personnel discuss cybersecurity, they are generally addressing the need to provide mechanisms (a mix of technology, architectural choices, company policies, and operational procedures) that ensure system confidentiality, integrity, and availability (CIA). *Availability* means ensuring the proper operational state for a business to operate—that is, the state of the computing assets, applications, networks, and data, whenever they are needed. In the case of SCADA, this includes the computing components and peripherals that constitute the system. Still, the concept goes further than that. It also means addressing essential communications (e.g., to the RTUs and any system that needs to provide data to or exchange data with the SCADA system) and essential data required by the SCADA system—such as relational database files that hold historical data for reports and system configuration files for the basic functioning of the SCADA system. For a SCADA system that needs 100 percent availability, this includes issues like the need for a redundant, fault-tolerant design of the system and its communications, the choice of potentially having alternative (backup) facilities, establishing (and following) proper system backup procedures, and even requirements like the installation of an uninterruptible power supply (UPS) to ensure power availability.

Integrity means ensuring that information displayed by and stored in the system is accurate, up to date, of known quality, and confirmed to have come from the correct sources and been processed in the correct manner (in other words, that you can trust the data). The concept also extends to preventing unauthorized modification of data or configuration information. It also means ensuring that the system itself (hardware or software) is not changed without going through the necessary validity testing and recertification processes. SCADA systems usually include some level of data validity checking and quality flagging at the point where field data are processed and placed into the real-time database table. Few, if any, systems make checks beyond that point because the data are refreshed continually with new data from the RTUs. Some systems include mechanisms for validating their configuration tables (e.g., a CRC code or some form of data check code), but few perform such checks on a regular basis. Most make such a check only immediately after reloading or modifying the configuration information or if an application crashes due to bad data. Thus, unauthorized and deliberate modifications to these tables could easily go undetected. The prevention of unauthorized system modifications

is usually relegated to user ID/password access controls. Simple password schemes provide only weak protection against a determined cyber attacker.

Confidentiality is the concept of ensuring that information, both stored in a computer system and transmitted over communication links to other computer systems, is kept from being available to unauthorized personnel or applications. One of the key ways in which information can be made unavailable to the unauthorized is to store it (or transmit it) in an encrypted version (as *ciphertext*), so that only someone with the proper knowledge (e.g., a secret key) can decrypt the ciphertext and recover the original data. Encryption and decryption technologies will be discussed later in this section. In the IT world, the mantra taught is CIA—confidentiality being most important and availability being the least important of the three objectives. For an industrial automation system, it is usually the reverse: AIC. Availability is essential, whereas few SCADA systems contain much confidential information—not to say that confidentiality isn't important. But it is important that the data they contain and provide to both operational personnel and applications have integrity.

From the 1970s up to the 1990s, most SCADA vendors focused on making their systems reliable, feature-rich, and as foolproof as possible. Little thought was given to making these systems cybersecure, mostly because they were isolated and because it didn't occur to most people that such systems might be the target of an attack. The events of 9/11/2001 demonstrated that the U.S. had enemies willing to expend great effort and expense and even to accept their own deaths, in order to hurt the country. Starting in the 1990s, and accelerating as the Internet became ubiquitous, criminal organizations and foreign nation-states began using the Internet to attack and compromise corporate computer systems. The motivations for this malicious activity ranged from theft of intellectual property and technology to stealing financial assets. But some of this effort was targeted at gaining footholds in systems critical to our national infrastructure, which includes many SCADA systems.

To protect our SCADA systems, we need to consider a combination of technology as well as 'smart' processes and procedures. Clearly, these systems need to be given physical protections, but at this point, we will focus on cybersecurity technology, particularly the application of technical countermeasures. Technical countermeasures tend to fall into three major categories: those that provide prevention against cyberattacks, those that provide detection of cyberattacks, and those that help to mitigate and recover from cyberattacks. Also, there are technologies that apply to the entire SCADA system as a whole and others that apply to individual system components. A good combination of both provides the best cybersecurity protection.

Remote Access/Connectivity

Everyone is (or should be) concerned about hackers or terrorists gaining access to their critical control systems and wreaking havoc. Having said that, hackers and terrorists are not magical creatures. They use the same technology as everyone else; they just put it to bad use, rather than good. (The term *cracker* is sometimes used to differentiate those who use hacking skills for evil; ethical hackers, using the same skills for good, are sometimes referred to as *white-hat hackers.*)

There are really only two ways to make a cyberattack on a computer system: you (or someone acting on your behalf) need physical access to the system, or you need electronic access to the system. If you have physical access to a SCADA system, you probably wouldn't be restricted to making a cyberattack, although that might be your preference, depending on your objectives and desire not to get caught. (A man with a fire axe in your server room can shut down your system even more effectively than a hacker with a cyberattack.) With physical access, you can also make use of the available human/user interfaces such as the keyboard/mouse/display and the various peripherals such as CD/DVD drives, USB ports, and Ethernet ports for launching a cyberattack. If you had no physical access, then you would need electronic/remote access to launch a cyberattack. In general, by electronic access, we refer to having a telecommunication link, via LAN (possibly wireless), WAN, or even telephone technology, that lets you communicate with the system (or some system service, utility or CLI). Hackers or terrorists who cannot physically gain access to your SCADA system(s) need to establish a communication link of some type to do their dirty work. A slightly less clear issue is the possibility of using removable storage media (e.g., a USB flash drive or CD) or portable computer-based devices (e.g., a cell phone or laptop PC) as a way to compromise your SCADA system. Clearly, those approaches will require that someone, not necessarily the attacker, will need to bring the media/device into the facility and introduce/connect it to the SCADA system. We will discuss an approach for getting that to happen (called social engineering) later in the book. Remote connectivity requires some viable communication pathway into the SCADA system that is accessible to the attacker. SCADA systems today no longer tend to be isolated because of the need to feed their information into a range of business and support systems, all of which tend to be on the corporate WAN (which invariably has a connection to the Internet). But there are ways to secure such connections. And the problem is compounded with the extension of corporate WANs out to field sites (a.k.a. *IP to the field*). SCADA systems today may also require communication interconnections with other organizations such as a regional authority, customers, suppliers, or regulators, and these connections can potentially provide an attack pathway as well.

Communication connections can come from many sources, can be permanent or temporary, can be over public or private WAN connections, or via an internal LAN connection (fig. 6–1)—with a highly interconnected SCADA system. Prior to the widespread establishment of the Internet, a temporary communication link would

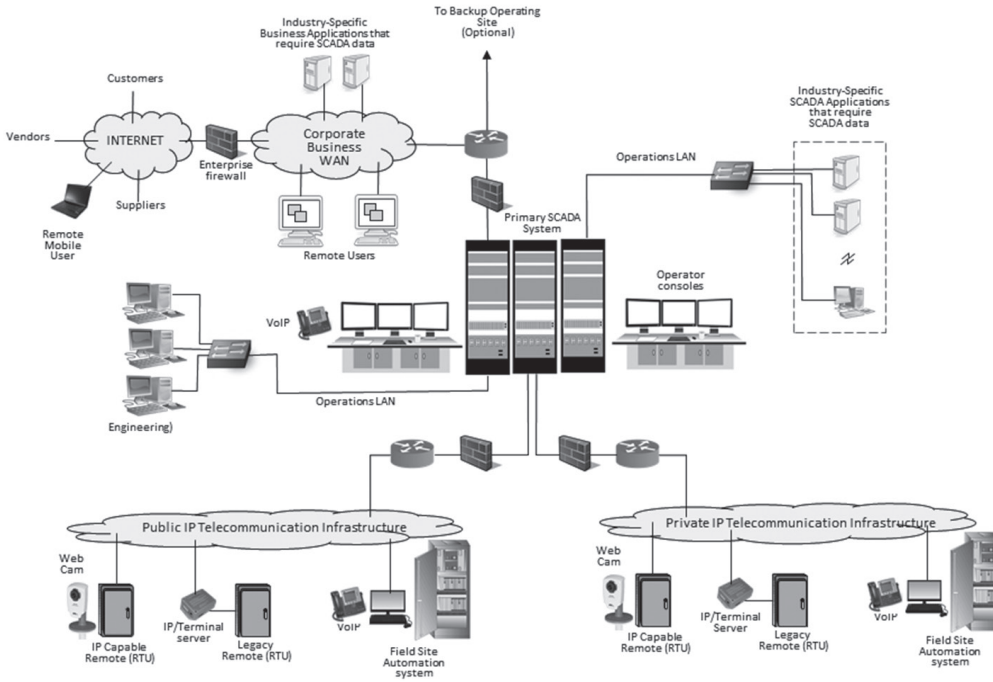


Fig. 6–1. Communication interconnections to a SCADA system

mainly have been established via analog telephone dial-up. Most SCADA systems used to have auto-answer modems and dedicated telephone lines that enabled the system vendor to dial in to the system, for troubleshooting and diagnosis of problems and installation of updates and patches to software. Today, such remote support, if allowed at all, would probably be via an Internet connection setup with a VPN gateway to protect and secure the connection. That connectivity might also require that a temporary "hole" be created in any firewall(s) that is (are) used to isolate the SCADA system from the Internet or the corporate WAN that provides the pathway to the Internet. The same would be true if there were a need to provide temporary remote access to the SCADA system for company personnel. One key aspect of SCADA cybersecurity is the elimination of unnecessary communication paths into the SCADA system and the placement of protective and detective measures on those that remain. A good starting point (even today) is to ensure that any such legacy modems and telephone lines are eliminated. Hackers developed programs called *daemon dialers* or *war dialers* many years ago, and such programs are still readily available on the Internet on various *hacker Web sites.* A war dialer program is given a fractional telephone number (e.g., the area code and first few digits of a phone number exchange) and then it sequentially dials all possible telephone number combinations, one at a time, recording what it finds. The program serves to look for modems, and when it finds one, depending on the program's sophistication, it may send a few test messages to see what response

is generated by the computer it has located (or to determine that it has just found a FAX machine). The response may tell the program what operating system software is running on the computer it has just reached. Later, at leisure, the hacker will review the information recorded by the war dialer program to decide if any of the computers should be further probed using other, more sophisticated tools. Your IT department can also employ such a tool to search for abandoned or unregistered modems within your own facility. (They can locate forgotten FAX machines as well!)

Another pathway that could exist is if any wireless Ethernet (a.k.a., Wi-Fi) access point (AP) has been connected on the SCADA system LAN (whether authorized or not). Although the official range of such an AP is usually just a few hundred feet, based on experience, using special antennas and amplifiers, hackers have successfully interacted with wireless APs from the distance of several miles. (They hold *war driving* competitions to see how many APs they can locate and break into and how far away they can be and still connect.) Today, if wireless Ethernet is being used on your operations network (and you really can't eliminate it for some good reason), it should at least be protected with suitable cybersecurity protections. WPA2-AES (IEEE 802.11i-2004) is currently the best protection available, but WPS (Wi-Fi Protected Setup) should be disabled in all APs, as it has known exploitable vulnerabilities. There is an even better security standard on the horizon called WPA3. WPA3 will protect against dictionary attacks by implementing a new key exchange protocol, to eliminate a weakness in WPA2. As of this date (mid-year 2020), some early products are being tested and certified, but full availability of products is expected to begin in 2021.

Clearly, an obvious pathway to the SCADA system would be the communication channels going out to the field sites to enable communication with the RTUs as well as other site equipment and systems. There has been a lot of discussion among various industry cybersecurity groups about what kind of damage could be done if an attacker were able to access such communication channels, possibly by physically breaking into a field site that is unmanned and remote. As we have already discussed, there are RTUs (and PLCs) that still use legacy serial protocols, as well as newer devices that support IP-network connectivity and IP-based protocols. If an attacker can gain access to a legacy serial circuit, they will typically be required to break that circuit (fig. 6–2) and connect a computer with appropriate interface hardware (e.g., a MODEM) to one side or the other of the break—i.e., connect to communicate with the host or connect to communicate with the RTU(s). If you don't break the circuit, then the polling messages from the SCADA system—or the RTU responses to the SCADA system—will get mixed with the attacker's messages and it will all be gibberish. Although test set software is readily available for a wide range of legacy serial protocols, the attacker would still have to figure out which protocol was being used and select the master or slave software version for that protocol. At that point, if connected to communicate with the host, the attacker's computer would need to reply to polls with slave responses that made sense to the SCADA master (were not rejected as bad) while delivering values/status information of a malicious nature. If

connected to communicate with the RTUs, then the attacker would again have to use the appropriate master software for that protocol and send messages/commands to the RTUs (those on this particular circuit) to control their outputs and (hopefully) operate field devices in a malicious and destructive manner. One thing that is clear is that, even if connected to communicate with the host, this connection does not provide a means for cyber compromising the host (e.g., delivering malware), as serial protocols are very rigid in their structure and design and all permitted message types are pre-defined. Anything that deviates (e.g., exceeds the expected or maximum message length) would be treated as a bad/damaged message and would be discarded. And since these kinds of protocols were typically processed a character at a time, attacks such as a *buffer overflow* attack, just could not succeed. In fact, at the first received byte that didn't match expectations, based on the protocol rules, the current message would be discarded as bad. To provide total protection against any attacker being able to utilize the serial communication circuit, there are link encryption devices that can be inserted into most serial communication circuits (one at each end) and which provide a reasonable level of cybersecurity and immunity from link abuse. The only way to use the serial link in that case would be to break into an RTU's enclosure (or field site) and make a connection to the encryption device.

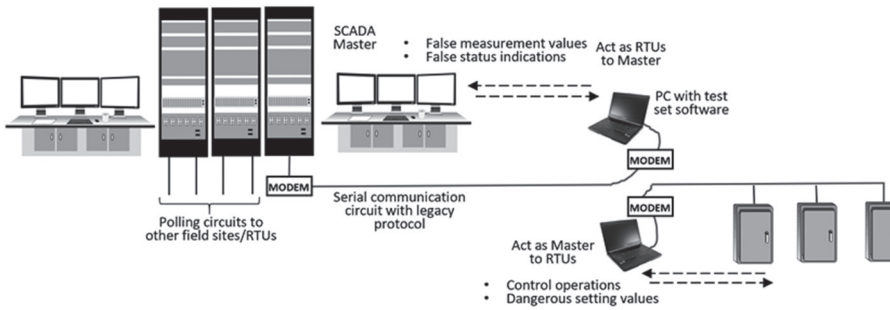


Fig. 6–2. Attacking and utilizing a legacy serial communication circuit

Unfortunately, the situation for IP networking to the field sites and IP-protocol-capable RTUs (or using a terminal server to provide a local serial connection) is quite a different matter. If an attacker were able to make a connection to that IP network, at any point, then in theory they would have a network connection into the SCADA system and could use it to attack and compromise that system. For that reason, if IP networking is used, it is essential to both protect the network with firewalls and to monitor the network for malicious, suspicious, and anomalous message traffic (fig. 6–3). It is also important that you take advantage of the security mechanisms built into things such as Ethernet switches, such as port security and locking. Since the RTUs should only be supporting a limited number of protocols and ports—one for polling but possibly also some for remote administration and support such as ssh, ftp, and even https—and should accept connections only from a specific set of IP

addresses, firewalls can be used to provide some level of protection, both for each RTU (or field site) and for the SCADA host computers as well. But since hacker tools easily allow for IP *address spoofing*, it is also important to monitor the network traffic out-to and in-from the field locations for anything unusual. A Network Intrusion Detection System (NIDS) sensor (possibly multiple) can be used to monitor and evaluate network traffic in near-real time and provide alerts if anything looks wrong, malicious, or suspicious. Depending on the manufacturer, a NIDS can provide signature-based detection for known malware, for known malicious message traffic, and *remote exploits*—and can also watch for anomalous message traffic (e.g., unknown IP addresses, unexpected port and protocol usage, etc.). Depending on how the NIDS sensor is positioned, it could even block pre-specified types of malicious traffic and prevent such traffic from ever reaching the SCADA system, i.e., acting as an intrusion detection and prevention system. In Chapter 11, we will discuss network intrusion detection/prevention technology. There are now firewalls that are protocol-aware of many standard industrial IP-based protocols (e.g., Modbus/TCP) and which can provide very specific and restricted filtering of those protocols.

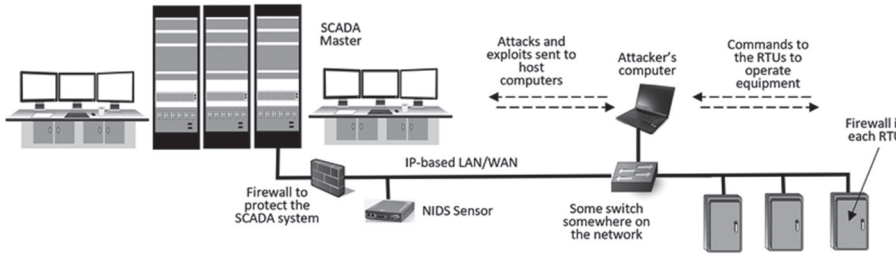


Fig. 6–3. Attacking a SCADA system that is using IP-to-the-Field

Today, with the prevalence of the Internet and the universal adoption of TCP/IP networking, telephone-based war dialing has been replaced by its IP equivalent: IP address *sweeping* (or *ping sweep*). If an adversary made a connection to your SCADA WAN using readily available tools, such as NMAP, they could scan and identify hosts on that WAN as well as open ports and services on each of those hosts. With IP networking, every computer on a network is given a unique IP address (like a street address), and a message containing a particular address will, when sent onto that network, be delivered to the computer with that address, if one exists. The postal system is a useful model for TCP/IP networking. People you don't know elsewhere in the world can send mail to your house, just by having your address. In fact, they don't need to know you or that this is your address; they just need to write down an address that looks valid, and if it happens to be yours, you get the letter. Valid IP addresses are much easier to guess than valid postal addresses (although there is probably a 100 Main Street in every downtown). But, as was already mentioned, most computer systems will never actually be connected to the Internet, and, in fact, most still use IPv4 even though the Internet

switched to IPv6 a while back. Truthfully, hackers can't actually see your computer or directly send messages to your computer across the Internet (and this is true at home as well as at work) because your computer is probably using a special type of "private" IPv4 address. Today, most corporations have their own WANs, and most still use IPv4 on those WANs as well as on each of their LANs. Because of the explosive growth of the Internet, it was clear that the available IPv4 addresses were going to be exhausted, and this was one of the reasons for rolling out IPv6. But the folks responsible for the Internet needed to provide some quick solutions to allow everything to keep working until IPv6 was read and deployed. One of the tricks they came up with was a special set of IP addresses that were only for private networks and which could not be used on the Internet itself (when the Internet was still running IPv4). Three (3) address ranges were established for this purpose:

10.00.00.00 to 10.255.255.255 (24 bits for host numbering)

172.16.00.00 to 172.31.255.255 (20 bits for host numbering)

192.168.00.00 to 192.168.255.255 (16 bits for host numbering)

Most home computer systems will be using the 192.168.xx.xx IP addresses (unless you have more than 65 thousand computers at home) and most corporate WANs and LANs tend to use the other two ranges because of the large number of the unique addresses they support. As mentioned, these addresses can't traverse the Internet, even when the Internet was still running IPv4, but they work just fine on a private IP-based network. So, your computer systems (including your SCADA system and Corporate WAN) are probably using IP addresses that are not reachable from across the Internet. So how can they ever be attacked (unless the attacker gains direct access to your private network)?

If a SCADA system is connected to any other system(s) via IP-based LAN or WAN technology, then the possibility exists that a hacker or terrorist could gain remote access to your SCADA system through those interconnections. Understanding how they accomplish this and what you can do to stop them is the main topic of the remainder of this chapter.

TCP/IP Suite

In the 1970s and 1980s, most computer vendors had developed their own proprietary operating systems and networking technology. This was fine as long as you purchased computers only from a single vendor. Unfortunately, that was not always possible or desirable, so the problem of *interoperability* (or the lack thereof) came to the forefront. It was always necessary to cobble together custom programs that could provide limited, hard-coded, and minimal data exchanges, when attempting to link computers from different vendors. Computers from different vendors had hardware architectural differences (e.g., the number of bits in a word and the bit orientation) and operating system differences (e.g., file structures, data representations, and command syntax).

During this same time period, a project was started within the U.S. Department of Defense Advanced Research Projects Agency (ARPA) to create a robust networking mechanism that would actually provide interoperability between and among all of the computer systems used by various government agencies, research laboratories, and defense contractors. The result of this project was called ARPANET (or DARPANET) and eventually evolved into the Internet we now know and love. The underpinnings of the Internet include a protocol suite of software (and the definition of a *virtual architecture*) that enables computers with different operating systems, file structures, data representations, and command-line interpreters to interoperate to some basic level. Within this software suite is IP (Inter-network Protocol), which is the protocol that actually carries all messages across the Internet (or IP-based private networks) in the form of IP datagrams. This is the protocol that utilizes the unique IP address assigned to every computer directly attached to the Internet, or on your own private network, to complete delivery. In addition, there are other protocols, like TCP, that improve the reliability of message transmission and allow messages to be directed to specific applications (or standard system services, like email) running on the destination computer. TCP "sits above" IP and performs activities to improve message delivery reliability. Because so many application programs actually make use of TCP for reliable delivery, it has become typical to refer to TCP/IP, rather than just IP. In all, there are dozens of protocols that are used to maintain, coordinate, and supervise the operation of the Internet. One of the important aspects of the Internet is that these underlying protocols are published standards that are freely available to anyone who wants to implement IP networking.

At about the same time that the Department of Defense was developing TCP/IP, there was ongoing work on an inter-networking standard by the International Standards Organization called the Open Systems Interconnect (OSI). But the emergence and overwhelming acceptance of the IP suite as a standard pretty much killed off OSI and eventually all of the various vendor-proprietary networking standards. This is not to imply that the OSI reference model was abandoned entirely. It is still used as a model against which other networking architectures—e.g., IP—are compared. Figure 6–4 compares the layers defined in the OSI model (7 layers) and in TCP/IP (5 layers). We will often hear IT personnel discuss a device as being associated with a given layer (e.g., a switch is a layer 2 device, a router a layer 3 device), and by this, they mean that their functions correspond to that particular layer of the OSI model.

Today, almost all intercomputer networking, regardless of LAN or WAN, is done with IP networking technology, aside from a few legacy vendor proprietary designs. (This is separate from smart industrial instrumentation buses—so-called *fieldbuses*—which do not use Ethernet or TCP/IP). A corporate WAN and/or LAN will employ IP networking technology, regardless of being connected to the Internet. In fact, such a stand-alone network may be called an *extranet.* If it is connected to the Internet, then the corporate network may be called an intranet (although in truth, those two terms never really gained much traction).

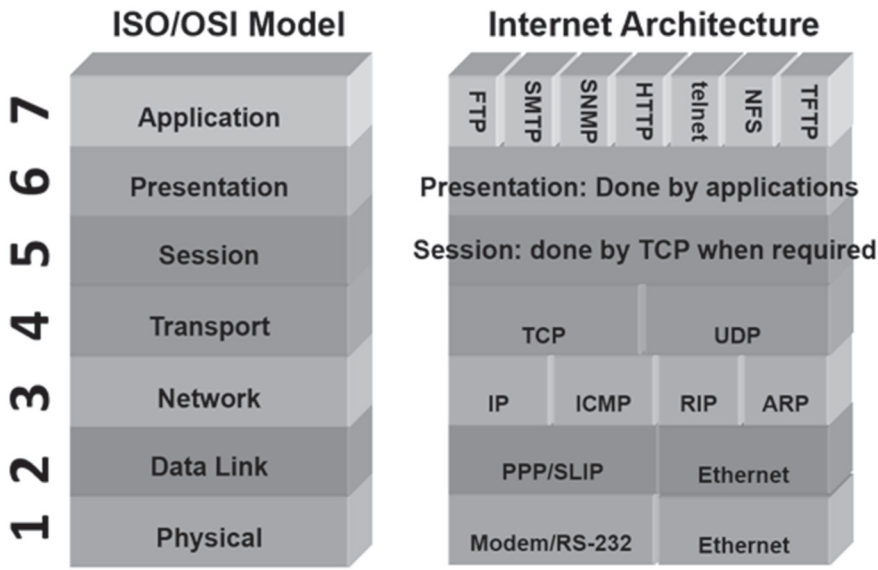


Fig. 6–4. The OSI seven-layer model and IP equivalent-function layers

The IP suite of protocols is provided with all of the popular commercial and open-source operating systems and is built into all of the COTS networking components (e.g., routers, firewalls and gateways). So, when we speak about a hacker attempting to gain remote access to a SCADA system via LAN or WAN network, we mean that the hacker is using the mechanisms of the IP suite to connect with your SCADA system. Since IP, as well as all other technical aspects of the Internet, are fully in the public domain, hackers have a vast amount of information to assist them in devising their attack methodologies. We have already discussed various physical WAN technologies such as FDDI, ATM, and SONET. All of these technologies have their own physical (layer 1) and data link-layer (layer 2) characteristics and hardware-addressing schemes, but in their link-layer frames, they all can carry IP datagrams—and there are gateways that allow these differing technologies to be interconnected so that they can pass the IP datagrams back and forth in order to get them delivered. Interconnected networks using SONET, ATM, and other similar technologies form the actual Internet.

IP addresses and gateways

As previously mentioned, all computers directly connected to the Internet today must have a unique IPv6 address (if they were previously connected with a valid IPv4 address, that address was converted, using a well-defined algorithm, into the corresponding IPv6 address). But for most of us, our personal and office computers are not directly connected to the Internet; we sit on a corporate LAN

or WAN and connect to the Internet via an ISP and their private WAN. (This is definitely how you connect to the Internet from home.) Only one computer on the corporate LAN/WAN may actually have an Internet connection (more likely that connection is probably via an ISP). The remainder of those computers must access the Internet by passing messages through this *gateway* computer. There are at least two good reasons for this architecture. The first reason is that most of the message traffic on the LAN/WAN is for destinations within the LAN/WAN, so there is no reason to pass them onto the Internet. The second reason is that the world was actually running out of unique IP addresses under IPv4, because of the unforeseen growth and spread of the Internet (rectified by IPv6). By hiding the computers on a LAN/WAN behind a gateway, we have the opportunity to reuse IP addresses (to use the reserved, private IPv4 addresses we have already mentioned). The computers inside the network all use special, private IP addresses that can never be sent across the Internet. These computers, if they need to send messages onto the Internet, send them to the gateway computer, which puts them onto the Internet, using its own IP address (which today is a real IPv6 address). (Later we will discuss how the sending computer decides if a message destination is local [on the same LAN/WAN] or not). In figure 6–5, we have a gateway computer that has a network interface on an internal, private network, and that interface has been assigned an IP address that is one of the IPv4 private addresses. However, that gateway computer has a second interface connected to the Internet, and on that interface, it has been assigned an actual IPv6 address. When a computer on the private network sends the gateway computer a message to deliver across the Internet, the gateway edits the IPv4 datagram and places its own IPv6 address in the datagram that goes onto the Internet. (Actually, it may just create an IPv6 datagram and put the IPv4 datagram in the data area of that new IPv6 datagram.)

The process of re-addressing (or repackaging) messages on their way in from, and out to, the Internet from a network, using private IP addresses, is called *network address translation* (NAT). The gateway computer (which might actually be a firewall or a router and not a "computer") performs this re-addressing for all of the other computers on the private network (fig. 6–5). Today, that includes packing or repackaging your IPv4 datagrams into an IPv6 datagram as well as lending you its real IPv6 address. The gateway computer may be the only computer on your entire corporate WAN that someone on the Internet can see. So, of course it would be a logical target of attack to establish a foothold in your corporate WAN. Often, this gateway is actually a router, and it could also potentially be your enterprise firewall. Actually even at the corporate level, your network may not have a connection to the Internet, just a connection to an ISP private WAN. In this case, there will still be a NAT function happening in your default gateway device because the IP addresses used on the ISP's WAN may be different from (or even the same as) those used on the Corporate network. A message may go through several NAT conversions on the way

through the Internet and back. Fortunately, the NAT function is transparent to you and your various applications.

IP addresses, subnet masks and default gateways

IP has an integral means for looking at a destination IP address and making a determination that this destination is either local (reachable on the same LAN or WAN) or not local, which means that some other means is needed to reach the destination. The gods of the Internet decided that a computer/device connected to the local LAN, and with an additional external network connection that might be able to get the message delivered, would be designated as the *default gateway*. On any LAN segment, if a computer needs to send a message to another which is not local, it passes the message to the default gateway and hopes a delivery can be made (and if no default gateway has been defined the message is just discarded as undeliverable). Originally, that gateway would have been the computer that was actually connected to both the Internet and the LAN. But the concept is now extended to any device that can pass messages elsewhere. The cable modem in your home is undoubtedly the default gateway for computers in your home (and probably has the IP address 192.168.0.1) but it can't reach the Internet, just the WAN of your ISP. But on that WAN, there IS a computer/router that CAN reach the Internet, and for your cable modem (on the ISP WAN), that is its default gateway. If you segment a network/LAN with firewalls, they usually become the default gateway for each LAN segment because messages for computers on the other side of the firewall can only be reached by passing through the firewall. If you have a router on your LAN that connects out to a WAN, it will be your default gateway. By convention, the default gateway device gets the lowest/first usable IP address in the sub-domain (e.g., for 10.XX.XX.XX/24 addresses, the default gateway would be address 10.00.00.1 and for 192.168.XX.XX/16 addresses that would be address 192.168.0.1). But how does IP make the decision that a destination IP address is local or not? When we discuss Ethernet later in the chapter, we will learn about a table in each computer that holds a list of local computers (called the *ARP cache*) and it would seem logical that this table would be consulted to make the local/remote decision. But Ethernet and IP were developed independently, and so IP does not assume an Ethernet LAN at the lower two levels (and, in fact, the lower two levels could be FDDI, ATM, frame relay, or some other networking technology). An IP (v4 in this example) address consists of two basic parts: a network number, which is assigned by IANA and is not alterable, and a set of lower bits that can be used to assign each computer (host) a unique number:

	Network Portion	Host #
Source IP	10011011110011100101110010100101011010011100110010	0100
Destination IP	10011011110011100101110010100101011010100111001011	1111
Subnet Mask	11111111111111111111111111111111111111111111	0000000000

The dividing line between those two parts was originally on an 8 bit boundary that could be determined by looking at the two most-significant bits of the overall address. But now it is identified using a second 32 bit binary number called a subnet mask. If all of the bits in the source and destination IP address match, for the 1 bits in the subnet mask, then the destination is supposedly local. If any of those bits don't match, then the destination is presumed to be not local and the message will be sent to the default gateway for delivery. This will happen even if the destination computer actually is local and could be sent the message directly. So, you don't want to mess up when setting IP addresses and subnet masks!

In IP networks, the source and destination IP addresses are stripped to the network portion (including any extra bits used for subnetting) and compared. Stripping is done by doing an XOR with the subnet mask. If the two stripped addresses match, IP assumes them to be local. If not, then the message is sent to the default gateway (if one exists). This means that if you mess up setting either your IP address or your subnet mask, communications might not function. Setting those up can be automated using DHCP services (dynamic host configuration), but in a SCADA system, it is usually preferable to use static IP address assignments. On a large IT network with lots of computers and constant changes, keeping track of who has what IP address can be difficult, and so DHCP was developed to allow a server to be set up to allocate IP addresses and track who was assigned which address. In each computer, you can specify if the computer will need to request an IP address when it boots up or you can manually set the *static IP address* (and subnet mask and default gateway address and DNS server address) in the computer's configuration. For an IACS (such as a SCADA system), it is generally recommended to use static IP addresses because new computers are not being added to the system and old ones are removed all the time, and because the failure of a DHCP server can't prevent a computer from getting an IP address or giving itself a totally different IP address. (A Windows computer set to request a dynamic IP address, and that can't reach a DHCP server, uses the Microsoft APIPA scheme—Automatic Private IP Addressing—to assign one to itself in the IPv4 range: 169.254.0.1 to 169.254.254.254 with subnet mask of 255.255.0.0). This can cause the computer to be in a totally different subnet and even block the computer from communicating through a firewall that doesn't recognize the address.

With real or private IPv4 addresses, the address includes some number of lower bits (below those designating the network number) that are allocated for giving each computer on the subnet a unique number. With a 4 bit host numbering scheme, for example, it would seem obvious that you have sixteen (16) possible addresses: 0000_2 to 1111_2. But, in fact, you don't. The all-zero host address (0000_2) is not usable or valid. The all-ones host address (1111_2) is considered as the IP broadcast address within that subnet. And, as matter of typical practice, the lowest usable host address (0001_2 in this case) is normally allocated to the default gateway

device on the subnet. (At home, your cable modem or DSL router probably has the IP address: 192.168.0.1.) Which means that with four bits, you can really only assign a unique host number to thirteen computers. You are not required to assign the lowest usable address to the default gateway (especially if you don't have one), but if you don't, then you need to ensure that everyone who can set address information is aware of that fact.

Phishing and spear phishing

Lately, a more common means for getting into your corporate network (and defeating your Enterprise firewall) is to use malicious email messages to get you to click on a hyperlink that causes your computer to make an outgoing TCP/IP connection to the hacker's Web site. At that point, the hacker can send messages back to your computer just like any other Web site to which you connect. In fact, the first part of connecting to a Web site (once a TCP session is set up) is asking the Web server to send back its home page (an http GET message), which in the case of a hacker Web site, would be an exploit aimed at compromising your Web browser. A Web site can't initiate a connection request to your computer (because your computer has no real IP address), but it can respond to one from you where your corporate (or ISP) gateway is providing you with NAT service. We will discuss a specific social engineering ploy called *phishing* (or *spear phishing*) later in the book. When you initiate an out-going TCP session to a web server (possibly by clicking on a hyper-link in an email message) your 'invisibility' is now compromised.

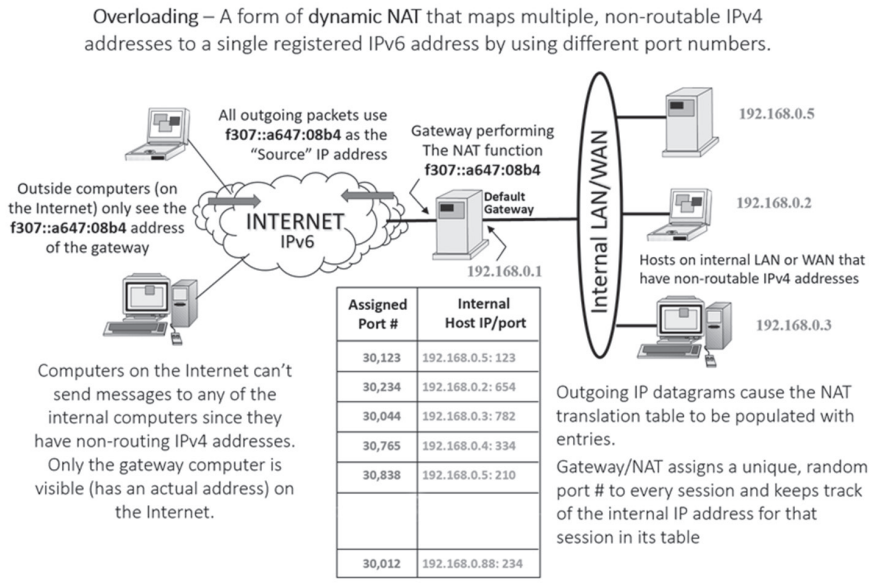


Fig. 6–5. Performing NAT in a gateway computer

We will be discussing IP, TCP, and UDP protocols (and even Ethernet) further, as understanding how they work helps to understand how hackers can abuse them and ways to protect them. Since most, if not all, of your networks are probably running IPv4, we will use that version as our reference throughout the book and only mention IPv6 where it is relevant to a specific cybersecurity issue. In our simplified example in figure 6–5, we showed your corporate network connecting to the Internet. In reality, it most likely connects to a private network operated by your ISP, which then connects to the actual Internet. So, your messages going to and from the Internet may actually go through a NAT process several times in each direction, all of which is transparent to you. We will make the point on several occasions in this book that a SCADA system should not have access to the Internet as this is opening up the gate to the potential attackers. It is fine to have cross-Internet connections, protected by VPN technology plus firewalls and NIDS sensors. but that is not the same as actually being able to web browse or send/receive email from the SCADA system.

Firewalls & Routers

In most cases in which a computer or a network (WAN/LAN) of computers is connected to the Internet, or even to another private network, the need exists for a gateway-type computer to deal with addressing issues and for ensuring that messages get moved along to their destination. The computer-device used for this purpose in most cases is a special-purpose computer called a *router*. This name describes its function: its job is to figure out how best to pass the messages along to get them to their destination, based on the IP address of that destination. It is necessary for that router to connect to at least two networks: the local LAN and the WAN that connects to—and provides communication pathways to—other locations. So, the router must have an electrical interface to the local LAN (e.g., Ethernet) and one to the external network (e.g., ATM or frame relay) A router must have at least two interfaces, but it may have many more. While the Internet was being developed, the designers used the term *default gateway* to describe computer functions dealing with addressing and message forwarding issues. (Today, we still use the term *default gateway* to describe the computer/device on our LAN to which we forward messages if they are not for a local destination.) The notion was that these routing functions would be assigned to one of the general-purpose computers on the network—that is, to the computer with one or more physical communication link(s) to the Internet (actually to other routers on the Internet—you really can't just run wires into a "cloud" and expect anything to work). Equipment manufacturers (such as Cisco) began building lower-cost, special-purpose computers programmed specifically to perform these functions and gave them the name router. If you have high-speed Internet in your home, the router provided by the phone or cable company, which connects to their system (their WAN) and provides you with high-speed Ethernet (wired or Wi-Fi), is your default gateway. If you are sending messages outside your home, then they

are being sent by your computer to that gateway box, which forwards them onwards (and probably performs the NAT function). Today, when we speak of a gateway, we are more likely speaking in particular about a router that is performing NAT functions (*as well as other functions*). Since *all* IP messages must travel through the gateway/router before reaching the other computers on any other given network, the gateway/router is the logical place to try to detect and block the entry of messages that are intended to harm any of the computers on the network. Routers today have the computing power and memory to support additional logic for the checking and blocking of IP messages (they can act as a *packet-filter firewall,* in addition to their other functions). Packet-filter firewalls perform very simplistic checks, based purely on the information contained within an IP message (*datagram*) header. All IP datagrams contain the IP address of the computer that is sending the message (the return address) and the IP address of the computer to which the message is to be delivered. Further, if TCP (or UDP) are used for delivery, there will be source and destination port numbers, as well as the IP addresses (fig. 6–6). An IPv4 datagram consists of an IPv4 header, followed by either a TCP, UDP, or ICMP header, followed by application data. The entire thing cannot be longer than 64 kilobytes. (An IPv6 datagram is similar in structure, but can be much, much bigger.) Messages traveling through an IPv4-based network will be carried, at the data-link layer, in frames appropriate to the underlying technology (i.e., Ethernet, ATM, etc.). On a WAN, the datagrams will be passed from router/switch to router/switch until reaching the one that can deliver the datagram to the destination computer. (Here, the term switch refers to the large backbone computers on the Internet that store and forward messages and not to Ethernet switches.) If the final router is your gateway, then—if it has packet-filter capability—it would inspect the data in the IP and TCP/UDP headers to make a determination as to allowing the datagram to enter your private network, or deleting it without passing it onward. This is a basic *firewall* function.

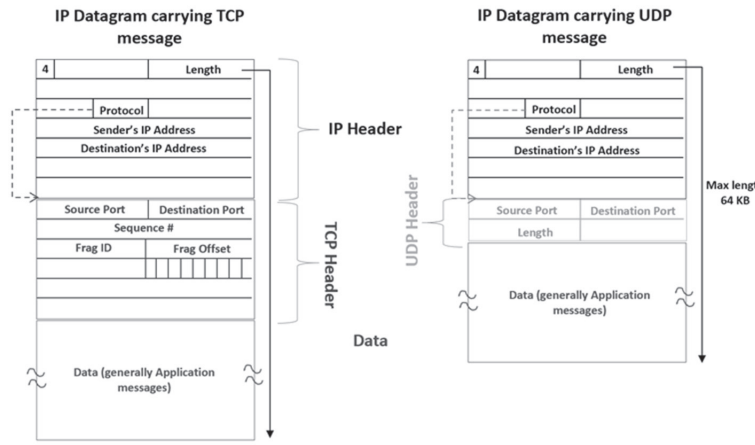


Fig. 6–6. IP and TCP (or UDP) datagram header information

This limited addressing information is insufficient today to identify message traffic that is harmful, both because malicious software can be hidden inside the message transported in the TCP/IP packets and because it is all too easy to fake the sender's address information. (The term *IP spoofing* is used to describe methods for creating messages with falsified source IP address information.) Hackers have become ever more clever in their methods for getting through firewalls, which is why no one depends solely on firewalls for complete protection of essential computer systems. Nevertheless, properly positioned and configured firewalls can be a good starting point for establishing an effective cybersecurity defensive architecture. Note that for an actual connection that is Internet-facing, it is highly likely that a much more powerful and sophisticated firewall would be placed immediately behind the router and it would provide much more advanced protections well beyond basic packet-filtering. Such *enterprise firewalls* (a.k.a., *next gen firewalls*) also offer support for malware detection and blocking; malicious Web site blacklisting; application traffic monitoring; SSH and SSL decryption, monitoring, and port control; VPN gateway functionality; identity-based security policies; stateful protocol inspection (SPI); and much more. Enterprise firewalls are essential for protection from Internet-based threats but are less commonly used within an organization's own private WAN and LANs. We will be discussing the different types of firewalls, including "industrial" firewalls, suitable for placement in the field, later in the book.

Because of underlying physical network limitations (such as the data area maximum size of an Ethernet frame), it is quite possible to have an IP datagram that is too big to send as a single, entire unit. (IPv4 datagrams can be up to 64 KB in total length, and standard Ethernet frames can only carry 1500 Bytes at a time.) Merely breaking a big datagram into smaller parts isn't workable, since the IPv4 header at the start of the datagram contains essential routing information (like source/destination IP addresses). Therefore, the datagram needs to be broken into "fragments" with each such fragment getting a copy (mostly) of the original IP header (fig. 6–7). Fragments travel across the Internet (or an IP-based network) independently and may arrive in a random order, so the IP header includes information (offsets) to help put all the fragments back together at the receiving end. Since firewalls (or NIDS) may inspect datagrams for malicious content, a hacker trick for avoiding detection is to break datagrams into very tiny fragments so that the malicious content is spread over several pieces (fragments) and thus is not recognizable. This scheme is used to bypass signature-based malware detection. More advanced firewalls will recognize tiny fragmentation and block such traffic as possibly/probably malicious.

The IPv4 header that is at the beginning of every datagram (or fragment) carries with it the IPv4 address (32 bits in length) of the computer sending the datagram and the IPv4 address of the computer to which this datagram needs to be delivered. This information will be used by routers/switches on the WAN to decide to which router/switch, or gateway, the message should be sent next. The header

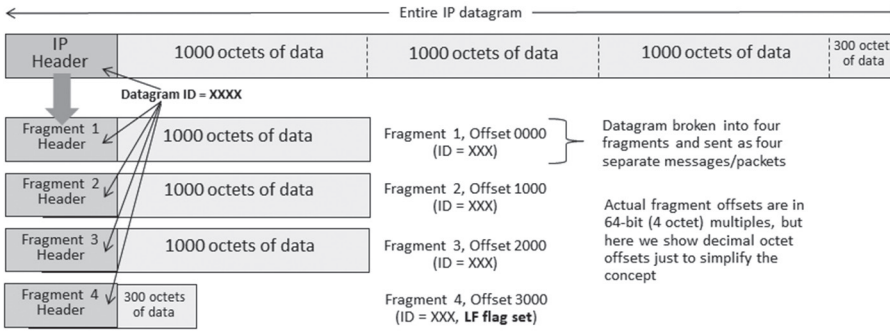


Fig. 6–7. IP datagram fragmentation

also has a "time to live" counter that is decremented by each router along the way (fig. 6–8). If the counter reaches 0, and the message has not yet reached its destination, the current router/switch handling the message deletes the message and sends an ICMP message back to the computer that originally sent the message, advising that the delivery failed. At that point, IP is done and takes no other actions to complete delivery. This is one example why IP delivery (and thus UDP delivery) is not guaranteed. Note that the deleted message may just be one fragment from an overall much larger IP datagram, and the other parts may actually get delivered. In IP, when a newly received datagram is identified (by its unique 32 bit ID number), a timer is started by the receiving computer (associated with that ID). If all fragments for that ID are not received before the timer expires, then the ones that were received are deleted and again an ICMP message will be sent back to the computer

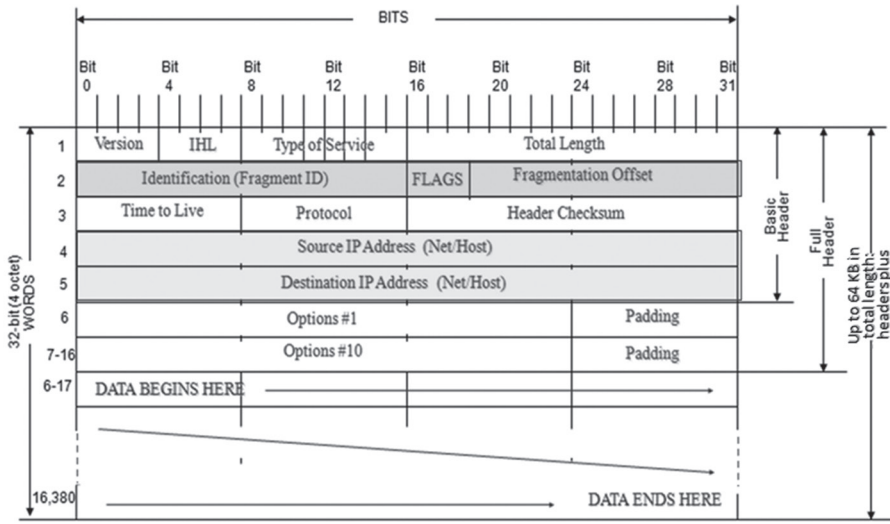


Fig. 6–8. The internal structure of the IPv4 datagram header

that originally sent the message, advising that the delivery failed. At that point, IP is again done and takes no other actions to complete delivery. One of the flag bits in the IPv4 header is the "final" bit indicating the last fragment of a datagram that was broken into smaller pieces. Note that fragments may not arrive in order, and so the first fragment received may not be the first piece of the datagram and the last fragment received may not be the one with the "final" flag set. The receiving computer allocates a buffer (up to 64 kB in size) and uses it to reassemble the datagram as fragments arrive. The fragment offset value (fig. 6-7) tells the receiving computer where in the buffer to place each piece. So, IP delivery isn't guaranteed, and datagrams may not be delivered. And that is why TCP was invented.

Cyberattacks against a computer system (SCADA or otherwise) can come in many forms and achieve very different results, depending on the objectives of the attacker. As was mentioned, there are criminal organizations, terrorist organizations, and nation-states all out across the Internet attempting to break into all sorts of computer systems for different purposes. The objectives in attacking a SCADA system could range from looking for competitive market information (e.g., in order to gain an advantage in bidding for power generation) or trying to use the system to damage the process (e.g., overpressure a gas pipeline, create a product spill on a crude pipeline, cause a significant and extended electric grid blackout). It could also be just to have a hidden future capability to inflict damage if there were a reason to need to do so (e.g., Iran declares war on the U.S. and knocks the power grid down to sow chaos). As Lenin said, "The purpose of terrorism is to terrorize." One of the reasons that SCADA cybersecurity is important is that much of the critical infrastructure of the U.S. depends upon SCADA systems.

Classes of attack messages

The classes of message traffic exchanged with a targeted computer as part of an overall cyberattack can generally be categorized into four broad types, based on the objective of the traffic:

- Messages sent to probe and explore a computer/network and identify hosts
- Messages sent to identify ports and services running on specific hosts
- Messages sent to identify and then exploit specific operating system, service, or application program vulnerabilities (and probably deliver an initial payload)
- Messages sent to deliver malware, tools, and other hacker software or commands once a computer has been compromised

Here, I am not specifically calling out high-volume message traffic generated to create a (distributed) denial of service attack ([D]DOS), as that message traffic is usually not, in and of itself, malicious. It is just that so much of it is sent to the target that resources are exhausted, and its services become unavailable to requesting clients (possibly because of crashing).

Probing and exploring

The TCP/IP suite includes a protocol with a set of message types whose purpose is the management and coordination of the computers on the Internet and traffic moving through the Internet. In particular, the Internet Control Message Protocol (ICMP) is used by computers to advise each other of situations when interconnecting communication links are overloaded and when messages in transit are discarded as undeliverable. Hackers have found creative ways to use these messages to gain information about the network and computers hiding behind the gateway. Every IP datagram has a *time to live* (TTL) counter, which is decremented as the datagram passes through successive routers/computers. If it reaches a value of zero, the router/computer with the datagram would delete it and send a message to that effect back to the originating computer. By sending a carefully crafted sequence of packets, starting with a TTL of one and with each successive packet adjusting the TTL value incrementally by one, it is possible to discover a lot about what lies behind a gateway/firewall. (This is how the *traceroute* command in both Windows and Linux operates.) Today, most firewalls give you the option of discarding and ignoring a range of message variations (including one arriving with a TTL=1 or a TTL < X where X is user definable) known to be used for such network probing. Readily available network assessment tools like Nmap (and Zenmap; refer to fig. 6–9, https://www.insecure.org) provide a number of ways of probing for computers and ports, even through a firewall, and identifying ports that are opened, closed, and filtered (blocked) by the firewall. They can use ARP, ICMP, TCP, and UDP message variations to do the probing.

Another type of probing consists of sending malformed IP datagrams (e.g., invalid flag combinations) or incorrectly sequenced datagrams (violating the rules of the *TCP state machine*) to specific TCP and UDP port numbers, to see how (or if) the system or service responds. If a service does respond, it may begin by sending its "banner" (the name, version of the software supporting the service). Tools like NMAP will perform *banner grabbing* and provide this information as part of its scan results. It is often possible to identify a great deal about the type and software version of an operating system, its installed updates and security patches, and the robustness of its security mechanisms by using such techniques. With this knowledge, a hacker can reference online lists of known vulnerabilities and attack methods, including those documented on the MITRE Web site in their ATT&CK database (https://attack.mitre.org). There exist readily available hacker (and security testing) tools that automate this process and generate a complete summary of identified, exploitable vulnerabilities. Your own IT group can use such a tool to identify vulnerabilities in your SCADA system computers. But the commercial version of these tools (a.k.a. *vulnerability scanners*) are very "noisy" (generate a lot of traffic), and their use would trigger alerts from most firewalls, NIDS systems, and the operating systems they are scanning. The hacker versions tend to offer a range of stealth modes and options that make detection much more difficult. We will discuss hacking tools a bit more in Chapter 11.

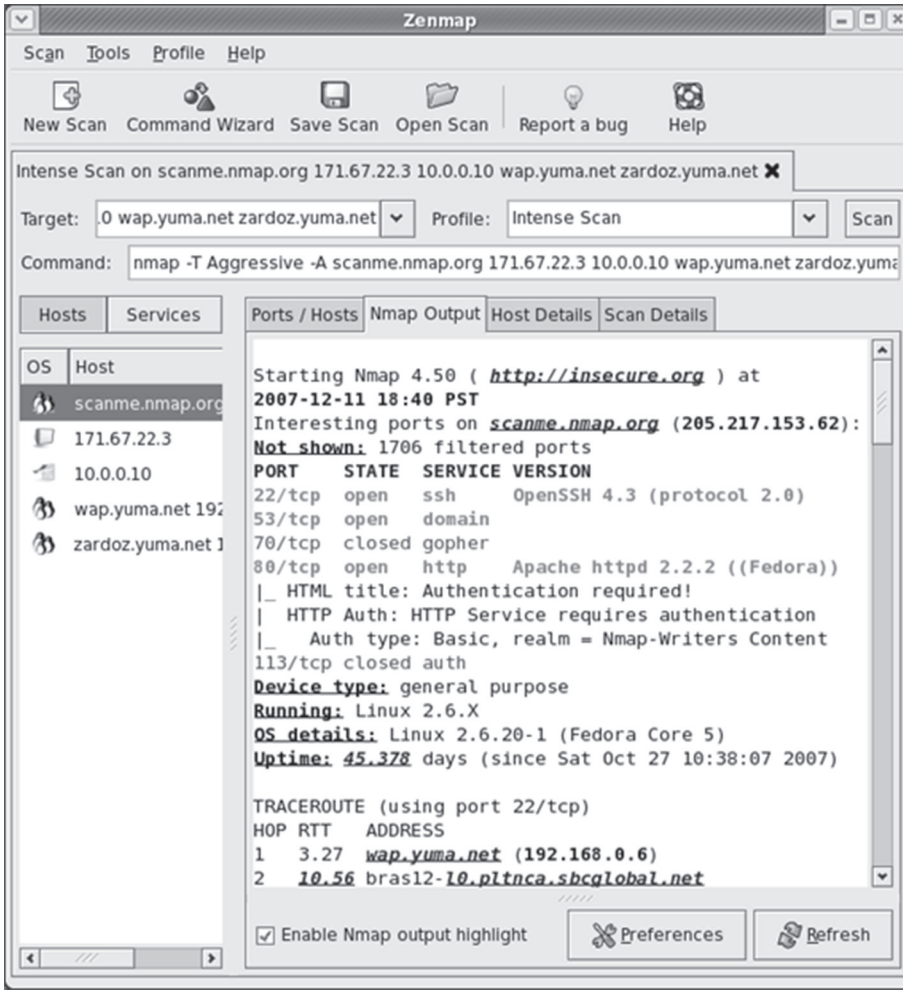


Fig. 6–9. The Zenmap network scanning tool

Exploitation

The kinds of remote exploit messages sent to a computer will depend on the intent. Computers do not have infinite memory, file storage, or computing capacity. It is possible to place an excessive and persistent load on a computer such that one or more of those critical resources is consumed. At that point, the computer system will either be forced to reboot or grind to a halt, ceasing performance of its required functions. An old cyberattack called *SYN flooding* exploited the failure of the IP implementation in most operating systems to keep track of how many TCP connection requests were made from a given remote computer or of how long it had been waiting for that initiating computer to complete the connection

negotiation. A hacker tool would send a continuous stream of TCP connection (SYN) requests (never fully completed), causing the target computer to fill its memory with storage buffers (one per connection request) until all memory was used and the computer operating system crashed. Today, all proper IP software implementations (and *stateful inspection* firewalls) can identify this type of attack and stop it before it endangers the computer system, usually by terminating the TCP session (and possibly by creating a dynamic firewall rule to block the sender's IP address).

A more difficult but similar attack is the *denial of service* (DoS) attack. This is where a computer (or its network connection to the Internet) is overloaded by having a large number of computers simultaneously send perfectly valid service requests (e.g., requesting a Web page) but at such a rate and quantity that all of the targeted computer's resources are usurped, thereby making the computer unavailable to valid users. Such an attack requires the participation of a large number of computers in cooperation, but hacker groups have successfully established huge collections of *zombie computers* (also called bot networks or botnets), spread all around the Internet, that they can use to launch DoS attacks (actually this is called a distributed DoS attack or DDoS). Modern firewalls can be configured with anomaly-detection technology to prevent the flood of messages from reaching any of the computers behind the firewall. (This would not necessarily prevent the DoS attack from tying up the firewall computer and making the Internet connection unavailable to the remaining computers.) DDoS attacks have been launched mainly against commercial Web sites (e-commerce and online gaming/gambling in particular), but any exposed (to the Internet or other untrusted WAN), unprotected computer system could be vulnerable to such an attack. This is another reason why a SCADA system should have no Internet access or connectivity.

Malware delivery

The term *malware* (malicious software) is used to describe a range of executable code introduced into a computer to alter its functionality. This could mean adding new (mostly undesirable) functions or eliminating, degrading, or altering existing functions—or merely just crashing the computer (which is the least useful result). The malware may perform a specific function such as creating a *backdoor* through which the attacker can have user access to the computer (e.g., a cmd.exe "DOS" command line interpreter on a Windows computer or a *shell* on a Linux computer) or it may be a *stager* that then brings over even more malware. There is a lot of opportunistic malware in the wild—malware that you "catch" usually due to poor cybersecurity procedures regarding portable media and use of email and Web browsing. This includes malware designed to encrypt your hard drive (a.k.a., *ransomware*), malware designed to seek credit card and social security numbers, and malware designed to capture user login IDs and passwords. The more serious threat is malware that is specifically selected and directed at your systems by an

active attacker who has done their homework. In this case, the attacker is generally trying to get a permanent presence in, or persistent access to, your system(s). Once the initial access is established, the attacker will take steps to establish *persistence* and to hide their presence. If the attacker can successfully install a root kit they will be able to make it extremely difficult for you to detect their presence, monitor their activity and to remove that malware. You may be forced to restore the system from backups in that case. The attacker may upload tools into the computer they have initially compromised and use it and use it as a platform to attack and compromise other systems (*lateral movement* to spread into those other systems). There are technical countermeasures that can be used to detect, and even block, such attacks and to make it very difficult to load and execute malware on your systems. In Chapter 11, we will be discussing *network intrusion detection and prevention* and *whitelisting* technologies.

Most SCADA systems now utilize conventional PCs (with Microsoft Windows operating systems) as their operator and engineering consoles. These PCs usually come with integral Ethernet, Wi-Fi, and even Bluetooth interfaces that are enabled by default. You can use all three interfaces at once, and if you do not have the correct settings, the PC could provide a bridge (perform routing) between these interfaces. So, a PC with a wired Ethernet connection to the SCADA LAN could communicate with another computer via its Wi-Fi interface and route messages from that other computer onto the SCADA LAN (fig. 6–10). A similar situation could occur if IP networking support were enabled on the Bluetooth interface. In both Windows and Linux, there are system settings that can disable IP routing functions, and this should be done, but the best approach is to also disable all wireless interfaces on any computer having a wired connection to the LAN supporting the SCADA system.

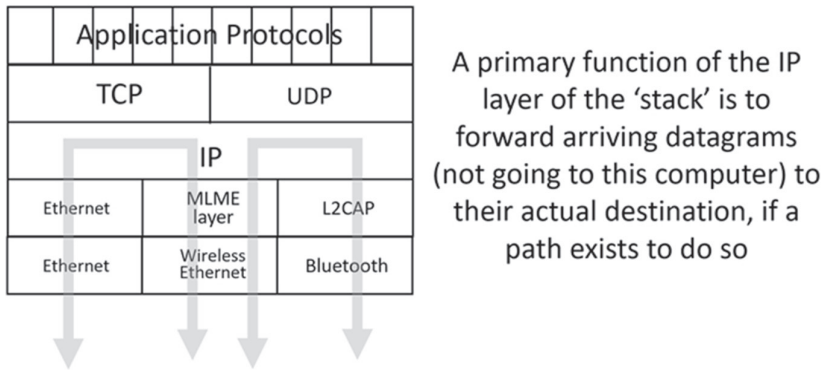


Fig. 6–10. IP routing between PC network interfaces

A good cybersecurity practice for computers connected to a LAN containing critical assets is to ensure that any wireless interfaces also installed on those

computers are disabled, preferably in a manner that makes re-enabling the interfaces impossible (physically remove the circuit cards for the interfaces) or at least very difficult (delete the driver software from the computer). Merely disabling these interfaces in an administrative manner makes it all too possible for them to be re-enabled, even unintentionally. For the same reasons, it is also usually considered a bad idea to install two (2) separate Ethernet NICs into a computer so that the computer user can access resources on two separate LANs, e.g., The SCADA LAN and the Business LAN (refer to fig. 6–11). It can be convenient for a SCADA engineer to be able to work on system problems and also able to check their email or Web browse without having to go to another computer, but that convenience creates an exploitable pathway. In addition, were the engineer to accidentally fall for a phishing attack and have their computer compromised, the attacker would have a direct connection from that computer into the SCADA system. Rather than allowing a dual-homed connection, it would be better, and more secure, to use a *KVM switch* to permit rapid switching between two computers: one on the business LAN and the other on the SCADA LAN. Just be sure to avoid mistakes like allowing portable media to be switched between the computers along with the keyboard, mouse, and video, as that could enable the delivery of malware to the SCADA system.

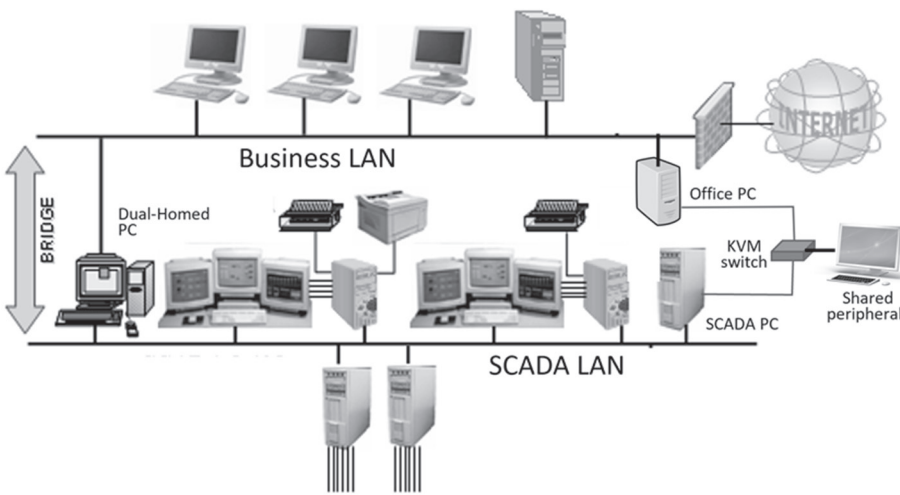


Fig. 6–11. Accidental bridging of LANs via dual-home connections

Another protective mechanism (and security policy) used by many companies is to have the Internet-facing firewall and email server keep a list of forbidden/dangerous Web sites (a list of URLs) and a list of spam sites (email addresses). Attempts to browse a forbidden Web site will be blocked by the firewall, as will incoming email from spammers. This particular protective mechanism, although intended as an aid to users, always seems to be a source of contention between the IT group and employees,

who find that legitimate Web sites and email are also being blocked. Also, even if attempts are made to keep the lists up to date, there will always be some amount of time lag. So, it is far safer to provide a second computer, connected to the business LAN, rather than allowing a dual-home setup. And a *KVM switch*, particularly a smart USB-based switch, can spoof both computers (they think they are still connected to the KVM peripherals) so that switching back and forth is seamless and instantaneous.

Exotic delivery mechanisms.

Up to this point, we have been discussing the delivery of malware to a SCADA system by means of remote electronic access, ignoring delivery by direct physical access using local exploits (which will be discussed later). However, most cell phones today, regardless of being Android, Google, or Apple-based, support Wi-Fi, Bluetooth, and cellular wireless connectivity.

These wireless communications technologies allow such devices to interoperate and exchange data with other similarly equipped devices. It is common to use Wi-Fi or Bluetooth to pass pictures, video, and other documents between a cell phone and a computer. It takes little imagination to realize that malware could be exchanged in the same manner. There are documented cases of malware specifically designed to be delivered to a Windows computer from a Bluetooth-enabled cell phone. Bluetooth is dangerous because, by design, it was intended to automatically detect and form *piconets* among nearby Bluetooth devices. Because that feature turned out to be exploitable, most cellphones and computers today require some level of manual intervention to make an association or *pairing*, at least the first time. But not always thereafter. File transfers via Wi-Fi are usually not as dangerous, because they usually require manual initiation. But a Wi-Fi pathway into any computer is an IP-network connection that could be used to launch an attack using remote exploits. This is just another reason to ensure that wireless communication capabilities are disabled on all equipment located on the SCADA network.

Ethernet LANs

Most systems today, IT or IACS, are built on top of high-speed, switched Ethernet LANs. The name 'Ethernet' is officially a Xerox Corporation trademark. The standard is formally IEEE 802.3, and through the years, this standard has evolved and improved and become the dominant LAN standard. Originally, Ethernet was a single, multi-dropped bus (cable) to which stations (computers) could be attached and could share that 10 Mbps LAN in a half-duplex, one-computer-at-a-time manner. With just a few computers connected to the LAN, Ethernet worked quite well. But as more and more computers got attached to the LAN, the performance would degrade due to increasing *collisions*. A collision occurs when two or more stations attempt to transmit at the same time. This is like multiple people trying to talk to you at once; all you hear is a mishmash of gibberish. Ethernet network controller

cards had a means for detecting collisions: they measured the peak voltage on the bus since multiple transmitted signals would combine and raise that value above its normal maximum. When a collision occurred, the stations would stop transmitting and then wait a random number of milliseconds and then try again (the assumption being that the colliding stations would wait differing amounts of time and thus not collide again). A few collisions were not a problem, but with too many computers sharing the bus, collisions became more likely and could then consume a good percentage of the available throughput. Another problem was that Ethernet frames (the basic set of bits that form a message sent over the bus) carried data destined for some program—but, like IP datagrams, only had an address for the sending and receiving computers. Nothing to indicate a program to which the data should be delivered. Figure 6–12 shows the basic structure of an Ethernet frame, although there are a few variations. The first two data items in all frames are the MAC (media access control or "hardware") addresses of the sending computer and the destination computer. These MAC (or Ethernet) addresses are 48 bit numbers, assigned at the factory, that must be unique on the given LAN. (The original concept was that every computer in the world with Ethernet would get a unique MAC address, but we ran out of them just like we did IPv4 addresses.) Ethernet switches use the MAC addresses to pass frames around, both within the same switch and out to other switches on a switched LAN. Original Ethernet frames had a 16 bit integer that indicated the number of bytes (length) in the data area of the frame, because Ethernet frames were of variable length. That field was taken over for another very important purpose (a type code to indicate what the frame was carrying in its' data area) when Ethernet version 2 was released.

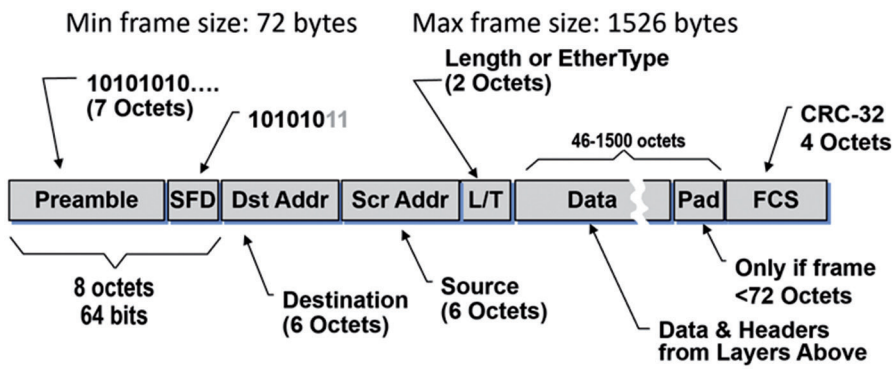


Fig. 6–12. Ethernet frame structure and contents

Every Ethernet frame has basically the same structure (except for "Jumbo" frames seen in Gigabit Ethernet implementations, which can be much larger, and tagged frames, which support VLANs and frame prioritization among switches). No frame can be smaller than 64 octets (72 including the preamble). This prevents

undetected collisions in maximum-size networks. The *Length/EtherType* field's function differs depending on whether you are using original IEEE 802.3 ("Raw") or Ethernet II (DIX) frames. (DIX—Digital, Intel, and Xerox—are the co-creators of original Ethernet.) Today, everyone uses Ethernet II, but very old equipment might still use original Raw frames. The EtherType field is very important because today, it contains a numeric value—most starting from 0x800 (hex)—and this number indicates what is being carried in the data area of the frame. Table 6–1 shows a list of a few EtherType codes you are likely to observe in LAN message traffic. Note that any value in that field that is greater than 1500_{10} (0x05DC hex) must be an EtherType code and not a length value because of the maximum data area size in the frame. A complete list of IANA-assigned EtherType codes can be found at:

https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml

Table 6–1. Ethertype values for some notable protocols

EtherType values for some notable protocols	
EtherType	**Protocol**
0x0800	Internet Protocol version 4 (IPv4)
0x0806	Address Resolution Protocol (ARP)
0x0842	Wake-on-LAN
0x22F0	Audio Video Transport Protocol (AVTP)
0x22EA	Stream Reservation Protocol
0x6003	DECnet Phase IV, DNA Routing
0x6004	DEC LAT
0x8035	Reverse Address Resolution Protocol (RARP)
0x809B	AppleTalk (Ethertalk)
0x80F3	AppleTalk Address Resolution Protocol (AARP)
0x8100	VLAN-tagged frame (IEEE 802.1Q)
0x8102	Simple Loop Prevention Protocol (SLPP)
0x86DD	Internet Protocol Version 6 (IPv6)

EtherType codes provide an invaluable capability—being able to determine what is in the data area of the Ethernet frame and thus being able to parse that data to identify useful information. (The maximum length of the data area in an Ethernet frame is 1500 octets, or 0x5DC, so a value of 0x800 or greater can't be a valid length; it must be an EtherType code.) For example, if a frame has EtherType 0x800, then we know an IPv4 datagram is in the data area. Knowing the structure of that datagram, we can locate the source and destination IP addresses. Knowing the location of the protocol field in the IPv4 header, we can see that a TCP header follows the IPv4 header, and so now we can locate the source and destination port numbers. This not only aids in handing multiple, concurrent IP message streams and TCP sessions, it allows us to deliver data to the applicable programs (based on those port numbers). In addition, this same modified functionality is why we can

have firewalls and network tools such as *Wireshark*, because without an EtherType code, they would be unable to parse the data area and locate specific information.

Switched Ethernet

Today, Ethernet LANs are constructed using devices called switches that are essentially powerful, high-speed computers with a large number of Ethernet NICs (ports) and a moderate amount of RAM. (You may still find Ethernet hubs, but they are essentially just a wiring convenience; they still allow collisions and force half-duplex operation; as switches became cheaper, hubs started going away.) Typically, except for low-end "unmanaged" switches, each NIC/port has a unique MAC address and some switches even have additional MACs for virtual functionality. They are usually diskless and use an embedded, specialized operating system that is "booted" into RAM from flash memory. Switches can be simple devices, with no advanced features, or complex devices, with a range of features, including features important to cybersecurity. In general, switches provide the ability to exchange Ethernet messages (frames) among and between devices connected to the same switch, or to other connected switches, but without the issue of collisions, since each device gets its own, individual, full-duplex connection to the switch and because the switch can buffer frames (up to a limit) until they can be retransmitted. The switch deals with speed and media differences and buffers frames in memory as needed (refer to fig. 6–13), but if there is so much traffic that the available memory is full, then frames will be discarded and lost (remember that delivery is not guaranteed with Ethernet, which is why we need protocols above Ethernet such as TCP). Ethernet does not perform retries or even confirm message delivery. It is assumed that these functions would be performed at a higher level.

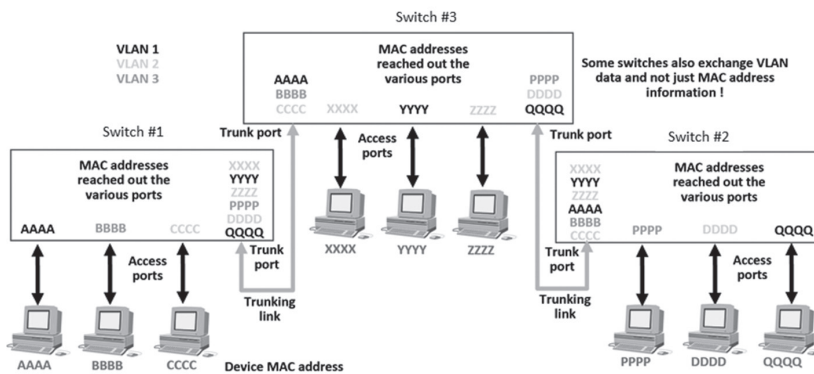


Fig. 6–13. Switched Ethernet LAN elements

Switches build up lists of MAC addresses associated with each of their ports (the MAC addresses of devices plugged into those ports), using the "source MAC

addresses" of incoming frames on each port. Also, when a frame arrives on any port, the switch examines the "destination MAC address" in the frame (the first data element in the frame), checks its address lists, and then knows which other port on the switch is to be used for the retransmission of the frame (which may be a port with a link to a totally different switch). When a switch first powers up, it has no MAC address lists, and so it *broadcasts* all frames to all ports (acts more like a *hub*) until it builds up the MAC address list. A switch is usually (but not always) just a level 2 device (per the OSI model), since it mainly deals at the link layer with MAC addresses. When, to get more ports or to connect devices to the LAN that are physically located elsewhere, multiple switches are needed, the individual switches are connected together with Ethernet cables, and the endpoint ports on each switch (the ports used for inter-switch connectivity) are (should be) designated as *trunk ports*. We will explain why this is important shortly. (Ports used to connect switches should be trunk ports; ports used to connect to end devices are called access ports.) Switches can be connected into a multi-level tree structure in this manner (fig. 6–13). In general, it is recommended that such a tree stay within four (4) levels, for speed and latency minimization purposes. (Two devices communicating, worst-case, have their frames passing through four switches going "up" the tree and then four switches going back "down" the tree, with each switch adding a time delay.) There is not a hard rule prohibiting more levels; it is just a general guideline. If you introduce too much aggregated latency time, some protocols or applications may think that delivery has failed and take some unnecessary recovery action. The problem with a tree architecture is that if any of the trunk connections between switches fails, the entire branch is disconnected from the tree and communications lost between assets. With a SCADA (or DCS) system, such a communication outage could be dangerous. It would be nice to have some means of providing a backup connection that could keep communications flowing. Ethernet makes use of several broadcast messages—messages that, when received by a switch, are repeated out every other port on the switch, including out the trunk ports to connected switches. (Broadcast frames can be identified because their destination MAC address is all 1s: 0xff:ff:ff:ff:ff:ff.) If you accidentally create a loop in an Ethernet LAN, you can create a situation whereby the broadcast message is recirculated around the LAN endlessly (a.k.a., a *broadcast storm*). In figure 6–14, a computer connected to switch A sends a broadcast message into that switch, and it gets sent to all other connected devices on switch A, including being sent to switch B. Switch B sends it to switch C, which sends it out to all connected devices, including back to switch A, and the cycle is repeated endlessly.

A broadcast storm can essentially consume all of the bandwidth of the LAN and the memory of the switches and cause a LAN failure. But because a switch or link failure can shut down part or all communications on a local network, some means was needed to allow for a backup/alternate path for critical communications, but without some new switch functionality such a backup path would create a potential loop and broadcast storm. Rapid Spanning Tree (RST; IEEE 802.1w) functionality

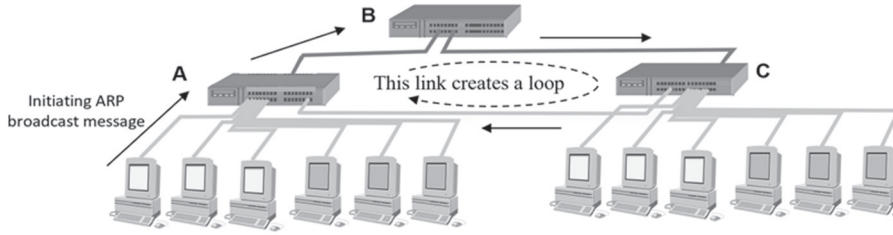


Fig. 6–14. Creating a broadcast storm in an Ethernet LApN

was added to switches to allow looped paths for automated failure recovery purposes, while preventing them from creating an endless cycle of circulating broadcast message traffic. With RST, switches constantly exchange special RST messages with other switches (via the trunk connections) to advise about their status and the state of their interconnections. A failure to hear "hello" messages from the root switch will cause a renegotiation of the various links and may enable links waiting in standby mode, thus restoring communications automagically (fig. 6–15). RST support can't fix a broken switch or switch port, but it can fix a lost/failed connection between two switches, as long as a backup/alternate connection is available. (If the LAN design doesn't include alternate/backup trunk connections, then there is no reason to enable RST.) RST is normally enabled in a switch on a per-port basis and normally only on trunk ports. RST is normally factory-disabled by default (since you don't know in advance what the individual ports will be used for) and needs to be enabled if needed. Older switches may have a specifically designated trunk port, but most current switches allow trunk functionality to be configured on a per-port basis.

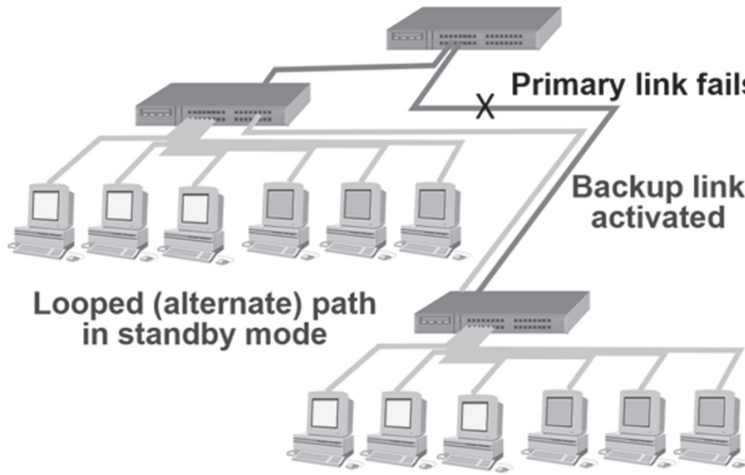


Fig. 6–15. Using RSTP to re-establish LAN communications

ARP cache updating/poisoning

In a system where computers are connected via an Ethernet LAN and use IP networking, each computer will have at least two unique addresses: a MAC address for the Ethernet NIC and an IP address for the computer. When you want to send a message across an Ethernet LAN to another computer, you need to know that computer's MAC address. If you only know its IP address, then you need to use a protocol called ARP (address resolution protocol) to get that MAC address. ARP messages are Ethernet broadcasts, so all computers on the LAN hear the message (ignoring VLANs for the moment). The computer with the IP address you specify in your ARP broadcast message will then send back a broadcast message telling you (and everyone else on the LAN) its MAC address. (These are Ethernet broadcasts and so they include the sender's MAC address in the frame, the IP information is placed in the data area of the frame.) Ethernet broadcasts allow all computers on the Ethernet LAN to hear these messages and use the data they contain to update their ARP cache (a table of MAC and IP addresses for local computers). Broadcasts may be passed by firewalls and switches, but normally not by routers. Broadcasts may also be restricted within a VLAN. One passive way that an attacker can gather information about the computers on your LAN (if they can get access to your LAN) is by listening for ARP messages (and to avoid detection, the attacker may disable ARP on their own computer). All the computers that can hear a given broadcast message are considered to be within the same *broadcast domain*. A common method used by hackers to insert themselves into LAN message traffic (and to allow them to hear all that traffic) is by performing ARP cache poisoning. This is a process of sending out fake ARP response messages that overwrite the valid entries in every other computer's ARP cache, essentially saying that the MAC address for every one of the IP addresses of the computers within the broadcast domain is the attacker's MAC address. (Each computer ends up with an ARP cache that lists all the IP addresses of the other computers, but the MAC for every entry is the MAC of the attacker's computer.) This causes every computer to send their message frames to the attacker, regardless of to whom they actually intended the messages to be delivered. Then the attacker can send the frame on to the intended recipient, possibly after making changes and definitely after inspecting its contents. This is a form of *man-in-the-middle* attack. Some switches can block ARP cache poisoning attacks using Dynamic ARP inspection (DAI). DAI protects computers against ARP cache poisoning. DAI performs validation by intercepting each ARP message and comparing its MAC and IP address information against the MAC-IP bindings contained in the DHCP snooping table built up in the switch. Any ARP messages that are inconsistent with the information contained in the DHCP snooping table are dropped by the switch. Of course this only works if DHCP is being used, which we have already said is not the best in a IACS/SCADA system.

Another capability of switches today is support for VLANs and differentiated traffic priority. Years back, Cisco heavily invested in streaming video and audio technologies (e.g., VoIP) and wanted to have a way to ensure that Ethernet frames transporting such streaming data passed through a switched Ethernet LAN with minimal latency

or time jitter (variations in network transit time). The IEEE developed a standard called IEEE 802.1Q/1p that allows switches to "tag" received frames with a priority (a.k.a. quality of service or QoS) and VLAN (virtual LAN) information, so that the switch (and other interconnected switches) would treat some frames differently. Frames received by a switch are immediately placed into an outgoing queue for the port to which the frame needs to be delivered, based on the destination MAC address. If several frames are received, going to the same outgoing port, they will be added to the queue. Waiting in a queue puts a time delay on frame delivery, since every frame ahead of your frame in the queue must be transmitted before your frame can finally be transmitted. The IEEE standard allows switch ports to be assigned a priority value between 0 and 7 (a three-bit value) where the greater the number, the greater the priority. As figure 6–16 shows, arriving higher-priority frames go to the head of the queue (or at least go ahead of lower-priority frames). Tagged frames typically exist within (and among) switches, and as a frame is sent out an access port, the tag is normally removed, unless that port is a trunk port and not an access port. This is why you must connect switches with trunk port connections—so that tags travel with the frames to other switches, where those switches can then also use the VLAN and QoS information. There are many vendor-specific variations on how tagging is done and how tags are treated and assigned, but in general, they are used in the manner just described. Some vendors assign VLAN and priority to ports based on switch configuration settings (e.g., access port 1 has VLAN 3 and priority 4, and any frame arriving on port 1 will be tagged with those settings). With this scheme, if I unplug your computer and plug mine into the same port, my computer's messages will now get the same VLAN/priority settings yours was getting (so now I am "in" the VLAN your computer was assigned to, which might be undesirable). Some vendors (such as Cisco) support a LAN-connected VLAN server that tells the switches what VLAN and priority values to assign each port, based on the MAC address of the device plugged into that port. In that arrangement, no matter what switch or port I plug my computer into, that port will then be set to the priority/VLAN values assigned to my computer and I can't break into another VLAN (at least, not that way—unless I change my MAC address).

QoS was added to permit time-sensitive traffic, such as digitized voice and video, to pass through a local area network with minimal time delay (latency) and jitter in order to preserve the quality of the delivered data stream (e.g., for VoIP phones)

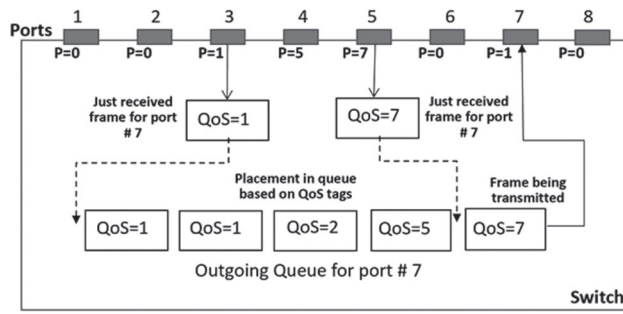


Fig. 6–16. Tagged frames with different priority values

Tag retention is why you need to use trunk ports to connect switches, so that tags remain as the frame is passed from switch to switch. The IEEE 802.1Q/1p Standards allow for adding "VLAN tags" and priority indicators to Ethernet message frames. These tags contain several elements. The Tag Protocol Identifier (TPID) is a 16 bit field set to a value of 0x8100 in order to identify the frame as an IEEE 802.1Q-tagged frame. This field is located at the same position as the EtherType/Size field in untagged frames and serves the same purpose. (Essentially, a tagged frame has two EtherType codes: one that says this is a tagged frame—in the position where the EtherType code is supposed to be located—and the original one set by the sending computer's TCP/IP stack, following the tag data; see fig. 6–17.) The Priority Code Point (PCP) is a 3 bit field which refers to the IEEE 802.1p priority. It indicates the frame priority level from 0 (lowest) to 7 (highest), which can be used to prioritize different classes of traffic (voice, video, data, etc.). The VLAN Identifier (VID) is a 12 bit field specifying the VLAN (number) to which the frame belongs. Most switches factory-default all their ports to VLAN 0 (zero) and priority indicator of 0 (zero) unless you change these values. The reason that tags are removed when a frame is transmitted out of a normal (access/service) port is that it would look like an invalid frame to the NIC card in your PC and would be discarded. There are hacker tools that allow an attacker to send tagged frames out of their computer and into an access/service port. Many switches will recognize the tagged frames, retain them and treat them according to the attacker's tag settings.

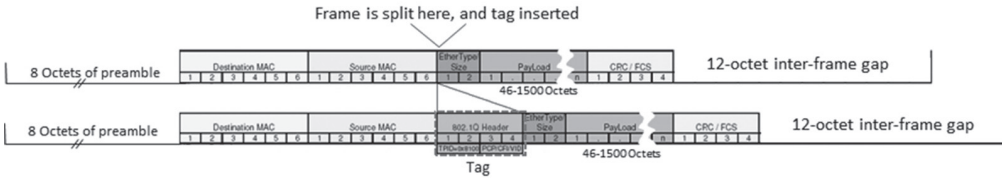


Fig. 6–17. Tagged and un-tagged Ethernet frames

One of the challenges with switched Ethernet LANs is that if you need to monitor a series of communication transactions between a pair of computers (a process also called *packet sniffing*, even though these are Ethernet frames and not packets), you normally can't, because their messages are passed through switches based on destination MAC address and so you won't receive them even if plugged into the same switch. The only messages you can see (receive) are broadcasts and messages sent specifically to your MAC address (and this is also why hackers use ARP cache poisoning tricks). Because the ability to monitor and observe message traffic is essential for debugging problems, switch manufactures generally provide the ability to set a *mirror port* on the switch (Cisco uses the acronym SPAN for this function), whereby you can replicate/copy message frames from any other port on the switch and have them sent to the mirror/SPAN port (fig. 6–18). Sniffing

Ethernet traffic (capturing frames) when a switch is involved requires using the port-mirroring or 'SPAN" port capabilities of the switch (or the use of various hacking tools to perform *ARP cache poisoning*). Although there are cases where we need to listen to all/most of the LAN traffic passing through a switch, it is far more common to need to listen to the message exchanges between two systems/devices that are having communication problems. In figure 6–18, two computers (A and D) are communicating and we need to see the messages for debugging purposes. By designating an unused switch port (#1) as a SPAN port, and telling the switch to copy port #7 traffic to the SPAN port, we can now use Ethernet sniffing tools, such as *Wireshark,* to capture and examine the messages being exchanged. Some switch manufacturers allow more than one port to be copied to a mirror port, as long as the switch has a spare port with adequate bandwidth. (For example, a switch with eight 10/100 Mbps ports and one 1Gbps port could use the faster 1Gbps port as a SPAN for all of the eight slower 10/100 Mbps ports.) At least one switch manufacturer also supports the ability to have frames sent to a SPAN port on a physically different switch in the overall tree (they call this an RSPAN for remote-SPAN). The ability to use mirror/SPAN capabilities of switches can be essential in solving communication problems, such as between an IP-protocol-capable RTU and a SCADA master. Note that with a SPAN port, the message traffic is only outgoing; you can't send message frames into the switch via a SPAN port. For that reason, a computer connected to a SPAN port is essentially invisible to all forms of network scanning. SPAN ports are seen as a security risk because a malicious insider could use them to capture sensitive information such as user credentials (in a system that uses centralized authentication) or SNMP passwords (community strings—except with v3 encryption). But fortunately, most switches will generate a Syslog message advising that a SPAN port has been created, and we shall discuss the value of Syslog messages shortly. Also, if a switch is being configuration-monitored (we will discuss this in Chapter 11), setting a SPAN port would be detected. One specific case where we want to copy ALL traffic going through a switch is to implement a network intrusion monitoring system (NIDS). This topic will be discussed later in the book as well.

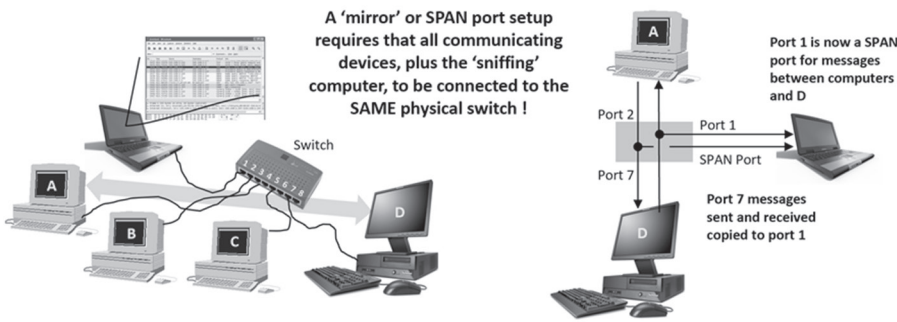


Fig. 6–18. Setting up a SPAN port on a switch

Switches can be a component of your overall SCADA system cybersecurity because switches today support a range of security mechanisms that can prevent unauthorized connections to your SCADA LAN and unauthorized changes to configuration settings, and can even provide communication security on your LAN. Most switches, except the most basic ones, allow for a function called *port security*. This is the capability to specify to the switch the MAC address(es) of device(s) permitted to be connected to specific switch ports. If someone tries to connect a computer/device with a different MAC address, the switch will refuse to accept and forward its Ethernet frames (and should generate a Syslog message). We have already mentioned that ports can be associated with a VLAN number. Switches won't deliver message frames tagged with one VLAN number to a port assigned to a different VLAN number. Although VLANs can be defeated with the proper hacking tools, in general they do allow you to segregate computers into functional groups, without having to use physically separate switches, and yet have isolation between the groups as if they really were connected to physically separate switches.

Some top-of-the-line switches support a capability called *MACsec,* which is essentially a VPN-like cybersecurity functionality that operates at the Ethernet link-layer level rather than at the IP-layer level like most VPN technology. This capability protects protocols that operate at the Ethernet level (and thus do not receive protections from conventional VPN technologies), such as ARP, RARP, RSTP, LLDP, and several others.

Another capability of modern switches is to provide some level of *network access control* (NAC) by supporting the IEEE 802.1x standard for *port-based NAC*. This is a function within a switch where the switch receives user authentication credentials from the computer through which the user seeks to access the LAN and assets on that LAN (called the *supplicant*; refer to fig. 6–19). The switch (acting in the roll of *authenticator* in this scheme) passes those credentials over to a centralized authentication server, possibly a Microsoft Active Directory server or RADIUS server, where the credentials are (or are not) validated. If the credentials are determined to be valid, the server notifies the switch that access should be permitted. Only at that point will the switch then allow messages (Ethernet frames) into the LAN from the user's computer. This same technology can also be used in wireless LANs, where a wireless access point (AP) performs the same authenticator function as the switch and blocks WAN/LAN access to the wireless supplicant unless authenticated. In fact, the use of IEEE 802.1x access control is highly recommended if you must have wireless Ethernet associated with a critical system. To prevent exposing user authentication information (e.g., Account ID and password), the transactions between the supplicant and switch/AP and between the switch/AP and the authentication server all use some variation of the *extensible authentication protocol* (EAP), which does not expose that information.

One final comment on switch capability and functions to support cybersecurity. Most switches, of any but the most basic level of functionality, also have the ability to generate, store, and transmit (using the standard *Syslog protocol* and message format)

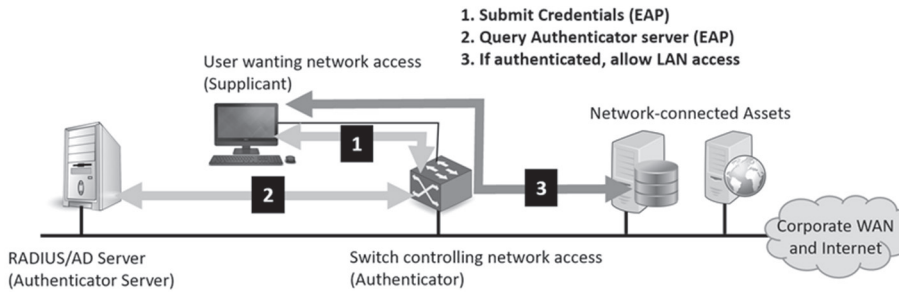


Fig. 6–19. IEEE 802.1x port-based NAC

time-tagged log messages when various events are detected by the switch. Connecting and disconnecting a device to a port will be logged, logging into the switch for remote administration (or via its console port) will be logged, making (most) configuration setting changes will be logged, rebooting the switch will be logged, and many, many, more events as well. These logs could be indicative of suspicious or malicious (or just unauthorized) activities on your SCADA LAN. They can also provide an unsolicited notice of a fault or failure. With a SCADA system, it might be reasonable to have syslog messages sent to the PC of the system engineer so that they are immediately informed about network changes, planned or unplanned. In Chapter 11, we will be discussing SIEM technologies for detecting cyberthreats and attacks. SIEM systems work best when they receive logs from as many time-synchronized sources as possible, including computers and workstations plus network elements such as switches, routers, and firewalls. All such network elements typically include a syslog client that can forward log messages to one or more syslog servers or destinations.

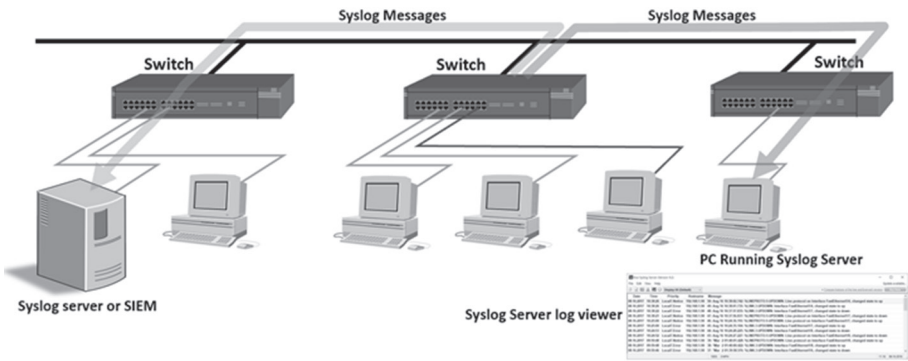


Fig. 6–20. Using Syslog protocol to send logs to a SIEM

Syslog messages are generally transmitted using UDP delivery to port 514 because no response or acknowledgement is expected from the server. A syslog message contains the following information:

- The date/time of the message generation
- The sender's IP address/hostname
- A facility code (source generating the message—system/process/application)
- A severity code ("debug" to "emergency")
- A vendor-defined message field

There are 24 pre-defined facility codes and eight pre-defined severity levels available to the product that is generating the log messages. Each vendor picks which values of each to assign based on the type of log message. Also, the content of the message field is totally left up the device manufacturer, with the only limitation being that a syslog message cannot exceed 1024 bytes/octets in total length. This creates challenges to SIEM systems that need to receive and dissect syslog messages from a wide range of products and vendors.

Wireless LANs

The issue of attack pathways into a SCADA system has already been discussed, and wireless local area networking (WLAN) is clearly a potential attack pathway, depending on how the technology is used and applied. Wireless Ethernet (a.k.a., Wi-Fi) is the technology of primary concern as it is deployed with IP layers on top of the wireless physical and data-link layers and thus offers full IP-networking capabilities. Wireless technology is a nice way of saying radio (although infrared links could be counted as well), and radio transmissions can be intercepted, blocked, and generated/spoofed. Although wireless Ethernet is too short-ranged to be used for communications to the field (from the SCADA master site), it would be very convenient to be able to use wireless Ethernet at a field site, for example, so that you can communicate to the site RTU or IACS while performing maintenance or calibration activities (fig. 6–21). It might be worth noting that there are industrial facilities that have taken this approach and who enable local access to the plant DCS by technicians and engineers. It can be very handy to be watching the control system's real-time readings on a wireless tablet or cell phone as you perform calibrations on field instruments. Similarly, within the SCADA master facility, it might be convenient to allow Wi-Fi access to the SCADA system. Wi-Fi is convenient, which is one of the reasons for its invention, but for cybersecurity purposes, there must be a proper balance between convenience and security. It is never a good idea to depend on wireless communications for safety applications or as the only means for communicating with critical systems or facilities. But it is possible to implement reasonable cybersecurity protections around wireless applications if you really, really need to use wireless. (This book is not going to deal with wireless instrumentation standards such as wireless HART or ISA 100—just with Wi-Fi, as those other standards already have extensive cybersecurity mechanisms built into their design, partially because of the IEEE 802.15.4 radio standard they both use.)

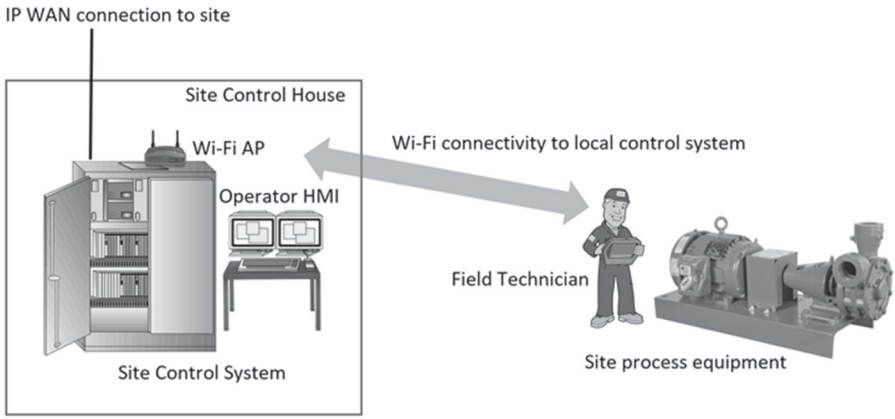


Fig. 6–21. Using Wi-Fi at SCADA field sites for local communications

The initial offering of IEEE 802.11 came with a very weak (and optional) set of cybersecurity protections called wired equivalent privacy (WEP). Today, as we discussed in an earlier chapter, we have much stronger mechanisms for encryption and authentication, mechanisms that essentially create a wireless VPN. After the security failure of WEP, the IEEE brought out Wi-Fi protected access (WPA) as a short-term fix, but it had some issues that make it vulnerable to compromise. The second version (WPA2), released in mid-2004, does provide pretty good cybersecurity, because it fully implements the IEEE 802.11i security standard with CCMP/AES (advanced encryption standard) encryption. Though more complicated to set up, it offers individualized and centralized control over access to your Wi-Fi network. Wireless clients are assigned login credentials they must present when connecting to the network, which can be modified or revoked by administrators at any time. Wireless clients never deal with the actual encryption keys. They are securely created and assigned per user session in the background after a user presents their login credentials. This prevents people from recovering the network key from computers (fig. 6–22). You also want to treat a wireless access point, and the clients that want to use it, as being untrusted and thus use additional defensive and detective measures. A good approach is to always place a firewall with advanced detection capabilities between the wireless AP and the internal LAN, which contains critical assets and systems. In addition, it would be advisable to monitor the traffic in and out of the WLAN using a NIDS sensor. WPA2 uses the AES encryption standard, which—for now, at least—seems to be immune to being broken. There are three variations based on the key size (number of bits) used: AES-128, AES-192, and AES-256. Authentication of wireless clients is just as important, and there are three possible approaches: using conventional user IDs and passwords exchanged using the PEAP protocol, using those plus an SSL digital certificate, or using TLS with digital certificates only. We will shortly discuss encryption, encryption keys, and digital certificates. But, to summarize,

it is possible to use Wi-Fi (wireless Ethernet) on your SCADA system LAN with a good level of assurance that you won't be opening up an attack pathway, as long as you apply the appropriate level of cybersecurity on your pathway. However, totally avoiding the use of wireless technology is preferable and clearly more secure.

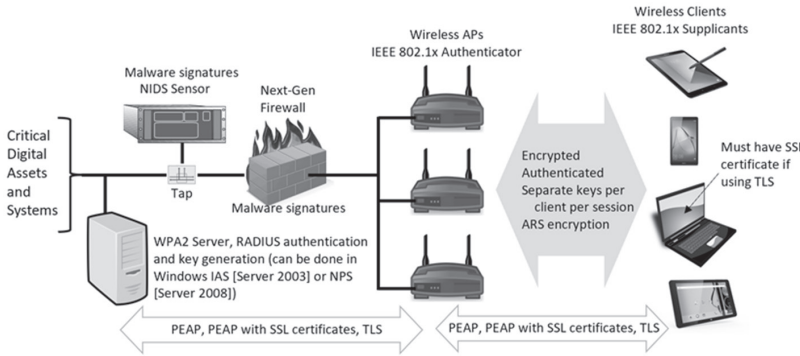


Fig. 6–22. Cyber security measures if Wi-Fi must be used

Authentication and Validation

When a person wishes to access the facilities and resources of a computer system today—whether that person is physically adjacent to that computer system or physically remote and making a cross-network connection—the procedure typically involves going through an authentication process of some sort. In most cases, this merely means entering a valid user ID and the corresponding password for a pre-existing user *account*. If the system accepts these two items as valid, then the person is granted the access rights corresponding to the account associated with that particular ID, as assigned by the system administrator. In this case, the password acts as a shared secret known to the legitimate user and the system they are accessing (or central authentication system), and it thus verifies that the user must be who they claim to be. Sophisticated operating systems (such as Windows, Apple's OS X and Linux/Unix) support some form of user account management that defines the access rights and limits for each user. Most users of a system will be restricted to running the applications to which they are allowed and working on their own files and documents. Accessing protected applications and system utilities or altering/deleting files belonging to others is not normally allowed to any user unless that user has administrative rights (the term *administrator* is commonly used in the Microsoft Windows world—like being the root or superuser in the Unix/Linux world). One of the most typical ways in which computer systems tend to be compromised, other than being subjected to local/remote exploits, is by the compromise and abuse of user (and/or administrative) accounts. Among the IT "best practices" for managing computer system accounts are the following:

- Assign the least amount of rights (privilege) required for the job/role
- Avoid sharing/shared accounts; give a unique one to each user
- Limit who has an administrative account; require special authorization
- Eliminate guest, temporary, factory-default accounts
- Set login failure limits and account locking
- Provide prior login date/time information at each login
- Require a periodic and/or event-driven review of all current accounts
- Use strong passwords (pass phrases) or (preferably) use multi-factor authentication

Access to the resources of a computer system (by personnel with an authorized account) can happen "locally" through the integral HMI of the computer (e.g., the display, keyboard, and mouse of your PC) or "remotely" (meaning via cross-network communications and protocols from another computer; refer to fig. 6–23).

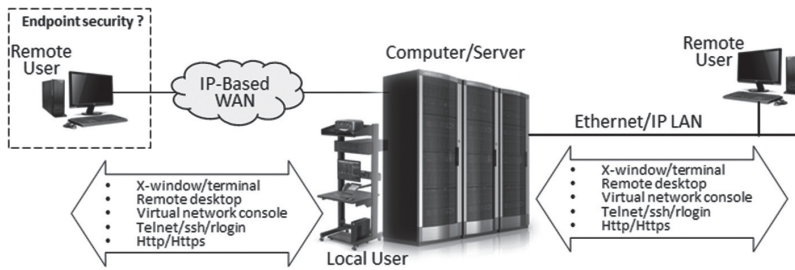


Fig. 6–23. Remote and local user access to a computer/system

By remote access, we do not mean inter-computer data exchanges, such as email transfers, Web browsing, and file transfers, even if those require providing a valid user account and password. We refer to gaining actual use of the computer/system resources as if you were locally *logged in* to the computer/system and sitting in front of the console and keyboard. Allowing remote user access to a SCADA system is dangerous and offers an attack pathway. Remote access via a LAN that is physically secured within a controlled facility is of less concern than allowing remote access via a WAN, especially the Internet itself, but still a concern, because there could be many secluded locations (e.g., wiring closet, telecom room) where an unauthorized connection to the LAN could be made if your physical security and/or switch security settings are lacking. This is of particular concern if your SCADA control center is in a shared office facility and not exclusive to the SCADA control center and associated functions. There are technologies for monitoring a LAN for the presence of new devices (i.e., doing periodic scans of the LAN), and a NIDS, for example, should detect Ethernet frames with unrecognized MAC addresses (but the MAC address of a PC can be modified) or unrecognized IP addresses (but those can be spoofed). This is why steps such as enabling port security and using IEEE 802.1x on your switches are important.

Remote access over a WAN is more problematic because an attacker can have numerous ways to get WAN access, such as using a phishing attack to gain a foothold in a corporate WAN (which happens all too often these days). Clearly, if the WAN is the Internet, then access is readily available. Unless other measures are taken to protect remote access, then an attacker typically just needs to compromise an existing account, possibly using *brute-force methods*. Many different mechanisms are available for gaining remote access into a system. Windows systems support remote desktop capability, which has been shown to have exploitable vulnerabilities, as has the use of Citrix and virtual desktops. Windows, as well as Linux, support telnet access, although that service must be enabled and is disabled by default. Linux systems can support X-terminal user access. Linux systems can support ssh and rlogin user access (and Windows supports third-party software such as OpenSSH and PuTTY). Both operating systems can support VNC (virtual network console) for remote user access. There are even Web-based tools (like RemotePC™) that provide remote access using http(s). And, in each case, the only thing blocking unauthorized use of these mechanisms to gain system remote access is the need for a valid user ID and password. As was mentioned, phishing one's way into a corporate WAN is often successful, and once in a corporate computer, the attacker can upload local exploits that will allow them to get a copy of the SAM file (the shadow file in Linux), which contains the user IDs and associated password hashes. Tools such as John the Ripper and the *SAM Juicer* in the Metasploit framework will eventually crack the passwords, and then the attacker has valid credentials for one or more user accounts. If a centralized authentication server (e.g., Active Directory) is used and *single sign-on* (SSO) is set up, then these account credentials can probably be used for remote access into multiple computers/systems. Cybersecurity will be enhanced if you proactively block/disable all remote access mechanisms except for the specific one you plan to use—and then implement a VPN connection for that mechanism, because that will require the remote user to have a valid digital certificate as well as conventional authentication factors.

Multi-factor authentication (MFA) is a good way to protect your user (and administrative) accounts from compromise and abuse. Multi-factor means proving who you claim to be with more than just a password. It is often discussed in terms of proving authenticity because of what you know, what you have, and what you are. Today "what you have" for MFA can include using separate devices such as an RFID ID card, your cell phone (receive a text with code#), or a hardware token (which generates a constantly changing one-time password [OTP] based on open authentication [OATH] standards), or a USB security key that supports Fast Identity Online (FIDO) standard (fig. 6–24). The "what you know" part of MFA can, of course, include a password, but also a PIN and even a challenge-response mechanism (e.g., answering pre-defined questions with the correct answer). The "what you are" part of MFA can be supported via facial recognition and fingerprint scanning or even the use of biometric scanners for hand geometry or retinal scanning (although those last two are more often used for physical access control rather than computer access). MFA is a very effective way to protect systems/computers from account compromise and abuse.

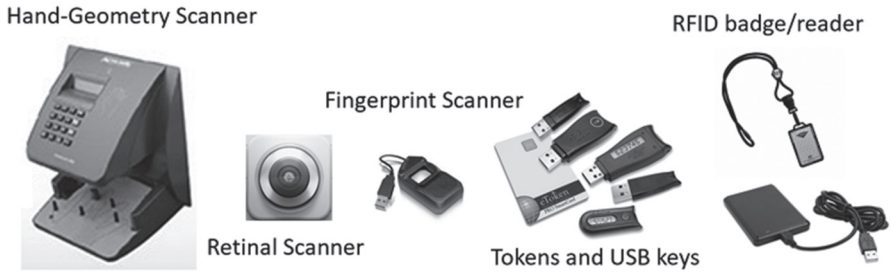


Fig. 6–24. Strong authentication options and technologies

The SCADA system operators often need to be provided with a different (or even no) authentication process. That is because they are not conventional users from the standpoint of a computer operating system. SCADA system operators actually interact with a suite of applications (the SCADA software) running on the computer system (in particular, the operator console and alarming functions). But they are not "users" in the conventional sense of the term (they are not going to be developing software, writing documents, performing engineering calculations and designs, etc.). For this reason (and for operational safety reasons), they are normally treated differently.

There are generally four categories of personnel who need access to the SCADA system:

- Operational personnel
- Engineering personnel
- Application programming personnel
- System administration personnel

A fact of life in some organizations is that staffing limitations force individuals to wear several hats and assume duties that fall across multiple categories. Sometimes a single individual wears all of the hats. That is a potential security risk, as you are giving someone all of the power needed to wreak great damage. An organization ought to make a business risk assessment and decide if the risk is worth the cost savings, or at least recognize the risk they are taking. (Risk assessment and business justifications will be addressed later in the book.)

The first personnel category (operators) consists of the people who sit in front of the operator display consoles—watching graphic displays, alarm summary lists, and historical trending displays—to supervise and control the process being monitored by the SCADA system. These personnel typically do not have a computer or possibly even an engineering background, but rather are specially trained to understand the process being monitored and controlled—specifically, to use the SCADA system displays and interfaces to manage and adjust that process. Because

these people have direct supervisory control over field equipment (e.g., circuit breakers, pumps, valves, and motors), there is often a subcategorization of access restrictions related to the training level and experience of the individual operational personnel (e.g., Senior Operator, Operator, Operator Trainee).

The second category of personnel (engineering) often make use of the data collected by the SCADA system for reports, calculations, modeling, and process optimization. These personnel usually have the responsibility for making configuration changes to the SCADA system (adding RTUs, database points, calculations, reports, etc.) as required. They usually have reasonable familiarity with the operational aspects of the system (often having been responsible for creating the displays that form the operator interface), although they do not necessarily possess the hands-on experience of the senior operational staff. The engineering staff usually have the most experience in the overall use of the SCADA system configuration utilities, communications systems, and computer hardware in general. To accomplish their work, these personnel must usually be given access to the full range of system configuration tools and the actual files that define the system functions and configuration. In this category would be the application/automation engineers, the telecommunications support staff, and even the hardware technicians that set up, install, and commission the RTUs.

The third category of personnel (programmers) may not be separate from the second, although in large organizations that is more likely. These personnel have specific programming expertise and have had to learn both the SCADA system—particular mechanisms for program development, testing, and integration—as well as the operating system utilities for the computing platform on which the SCADA system is built (e.g., Windows). To develop suitable application programs, these personnel must also understand the details of the system configuration (e.g., point tag naming schemes) and the process-related requirements (e.g., open a recirculation valve prior to starting the pump). This knowledge usually comes from interactions with the operational and engineering personnel. To perform their jobs, these personnel often need to be given access to the basic operating system utilities and functions, including the right to delete, alter, and replace the program files of the SCADA system.

The fourth and final category (administrative) is for the personnel assigned to perform the regular, routine, highly necessary system housekeeping functions—such as administering user IDs and passwords, making system backups, installing vendor-supplied software updates and patches, purging old files and old data from the system (possibly after transferring these data to removable media), and reviewing system logs for indications of developing problems. In smaller organizations, these functions may again be assigned to one of the engineering or programming personnel, in addition to their other duties. System administrators must be given access to system backup media, allowed to replace critical system files and software, given the ability to create and modify user accounts, and even granted authority to shut the system down and reboot/restart the system (with proper coordination with the operational and engineering personnel, of course).

It should be apparent that each category of personnel warrants a somewhat different level of access and authority and that proper training and care are critical for those positions that include access to system functions that have the potential to disrupt or disable the system if misused. In most SCADA systems, the particular ID assigned to a user will define the access rights of that user (a.k.a., role-based access)—and as we have seen, some users must be granted very high levels of access.

Since a computer is just a machine, it has no direct way to confirm that the person sitting at workstation 3, who just logged in with the ID and password of the senior system administrator, is actually that person. To confirm this as true prior to granting full senior-operator access privileges, the computer must make the person provide some proof. If it were a human guard, the computer would ask for a photographic ID. As a computer, the type of proof must be something possible through programmatic means. The usual mechanism is to have the person provide a secret known only to the person and the computer. That is what a password is supposed to be—a shared secret. Unfortunately, most people use very simple, easily guessed passwords. Worse, many SCADA systems come with a set of default passwords, and the customers either never change them or change them once and never again. Worse still, many SCADA system vendors used to install a secret factory ID/password pair (granting full, unrestricted access) known to all of their current (and former) employees—and it is usually unchangeable by the customers.

It is common and typical to see operator consoles on a SCADA system on which a senior operator had logged in once upon a time and which has remained logged in ever since, leaving senior-operator access privileges available to anyone who can physically or logically access that console. It is also not uncommon to see IDs and passwords posted in obvious places (frequently on Post-it™ notes stuck to the CRT display) or the adoption of a single ID/password that everyone in the control room knows. If no remote access to systems were possible and if all (current and former) employees could be absolutely trusted, then authentication considerations might not be an issue. In that case, the major reason for such IDs and passwords would be to prevent or reduce the number of human errors. Unfortunately, these days we really do need to ensure that only those with proper authority gain access to critical automation and control systems, particularly if they are remote users making an electronic connection. Fortunately, in the situations just described, the problems can be addressed through training, education, and the establishment of appropriate policies and well-crafted procedures, Actually, it is surprising how much progress can be made, by these same measures, toward reducing vulnerabilities and the consequences of an attack, without the need for technical solutions. Account management is a serious issue in an IT operation that may involve many hundreds of users and many dozens of systems especially as personnel and their assigned roles may be changing constantly. But in a SCADA operation it is likely that the staff requiring user access will be a small and mostly stable set of personnel and thus account management and supervision is less difficult although no less important.

Encryption and Ciphers

As mentioned at the beginning of this chapter, two of the objectives for cybersecurity are confidentiality and integrity. SCADA systems typically only contain information regarding the process they monitor and control, as well as related information such as model-based calculations, schedules, operational plans, etc. It may not seem like this sort of information really needs to be protected. However, in specific industries and applications, some of the information in a SCADA system might be considered as business confidential, and there could be legal implications or competitive advantages gained if this information were released. In the deregulated electric power industry, a lot of information regarding generating plants—their operational status and outage schedules—is considered to be confidential and competitive in nature, and disclosure of this information could cause market irregularities and lead to legal action. With a free market in electric power, knowing in advance, for example, that a plant was scheduled to be taken out of the market for maintenance could provide an unscrupulous trader an illegal advantage. In the pipeline industry, knowledge about pipeline capacities, current product delivery scheduling, and even facility operational status and other such information could be of value to a competitor, to a person playing in the futures market, and to a terrorist seeking a sensitive place to create an explosion. Although we have focused so far on the threat posed by a terrorist or hacker, it is also worth remembering that another cyberthreat could be posed by a competitor (who might pay a hacker to break into your systems) or criminals seeking financial information.

One of the best ways to protect information, both stored in a computer and during transmission between computers, is to make that information useless to people who don't have the right to access that information. Encryption is the (reversible) process of running messages (documents, RTU polling commands, data exchanges, etc.) through a mathematical process that converts them into something incomprehensible and then, at a later point in time, translates them back into the original message.

It is worth stopping for a moment and remembering a basic truth about computers and digital communications: *everything* stored in a computer or transmitted/received by a computer is represented as binary numbers (0s and 1s). Regardless of whether you are trying to store or transmit a textual document, a graphic image, an email, or a video clip, all of these are stored as a series of binary numbers either inside the computer or, when transmitted, across a communication channel. That said, encryption can be explained as the performance of a series of mathematical operations on binary numbers to obscure their original values (a.k.a., to create *ciphertext*); then, to get back to the original numbers (a.k.a., *plaintext*), these operations are reversed.

As an overly simplistic example, you could choose to multiply all of the numbers making up a message by some other number, say 27. The person receiving the scrambled message would just have to divide all incoming numbers by 27 to

recover the original message. Of course, this means that both of you must know that 27 is the value used to (un)scramble the data. In the world of encryption, that number is called the *encryption key* or simply the *key*; the unscrambled (original) message is called *plaintext,* and the scrambled (encrypted message) is called *ciphertext* (fig. 6–25). Of course, an encryption scheme that just multiplies by a small numeric value would not take long to figure out (*breaking the encryption,* as it is known). In the real world, keys tend to be really big numbers (at least 128 bit values; remember that keys are stored as binary numbers as well), and the encryption schemes (ciphers) involve a lot more than simple multiplication (in fact, multiplication and division would not work for encryption because they are not reversible operations in a computer due to round-off). The basic objective of encryption is to make it take a lot of time and effort to break the encryption, preferably so great that the effort takes so long that the value or usefulness of the recovered information will be negligible or at least have diminished significantly. For example, if it will take a year to break the encryption on a message about a delivery schedule, it isn't worth the bother to attempt to break the encryption since the information is obsolete by the time the attacker has recovered it. Encryption involves two factors: an encryption algorithm and a key to be used for the encryption and decryption. As was mentioned, basic arithmetic operations don't work for encryption. Instead, Boolean operations such as doing a logical AND, XOR, OR operation, or performing bit shifting and bit rotating, are used because they are reversible and because they are part of the basic instruction set of all digital computers. Several of those operations require two binary numbers, and this is where the key comes into play. It provides the second binary value for the AND, OR, and XOR operations.

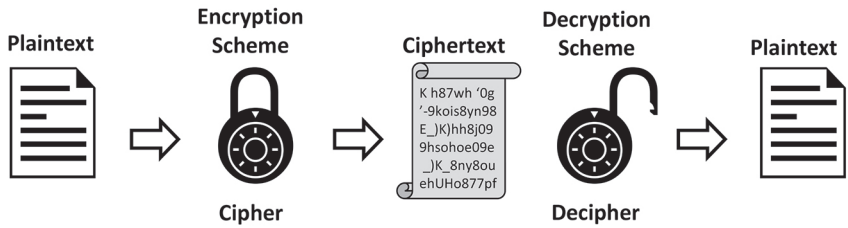


Fig. 6–25. Encryption and decryption of a document

Since an encryption algorithm (a.k.a. a *cipher*) must be reversible to be of use, there are actually not all that many of them, and they are all well-known since they had to be put up for peer review to ensure that they were workable. Some of the commonly used algorithms are: 3DES (triple DES), AES (advanced encryption standard), Blowfish, Twofish, and RSA (for public key encryption). These are called *block ciphers* because the way they work is to take a block of the data/message/file being encrypted and perform the conversion to ciphertext and then repeat the process on the next block till all of the data/message/file has been processed. Ciphers

are either designed to use *symmetric keys* (the same key is used to encrypt and decrypt) or *asymmetric keys* (one key encrypts but another, related, key decrypts).

Block ciphers are not workable for streaming data such as voice and video since you can't generate a block of either, wait for it to be encrypted, then transmitted, and then decrypted, and have it sound or look intelligible. For that reason, a different form of encryption was devised called a *stream cipher*. Your cell phone and desktop VoIP phone and Wi-Fi use a stream cipher to encrypt your conversation. Wi-Fi communications between the AP and wireless client are protected with a stream cipher. Several online video conferencing services use stream ciphers. A stream cipher takes each bit about to be transmitted and either flips it to the other binary value or leaves it alone (fig. 6–26). The decision as to which comes from exclusive-OR'ing (XOR) the bit to be transmitted with a bit generated by a pseudo-random number generator (PRNG) algorithm. The process is reversed on the receiving end, and both sender and receiver must use the same key value and PRNG algorithm/cipher. Stream ciphers in common use include ChaCha, RC4, Helix, Pike and SOBER.

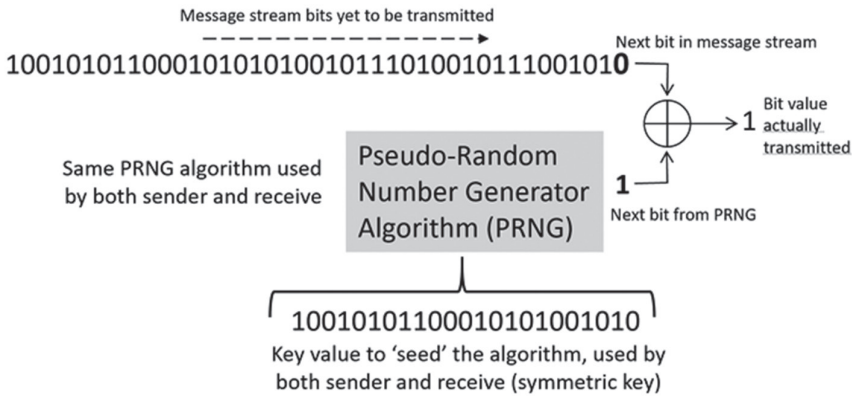


Fig. 6–26. Using stream cipher to protect transmitted information

As we have mentioned, encryption takes two elements: an algorithm (all of which are well known) and a (secret) key, which is then the most important factor. Keys are a shared secret, and you never want to disclose a key because with the key an attacker can decrypt your data and messages. Keys are just binary numbers, but usually of a significant length such as 128 bits or 256 bits. But why such as big number? The reason has to do with how keys are "broken" (their value determined). There are a range of approaches for determining a key value based on analyzing traffic and such, but a common method is just to try every possible value until you find the correct one. With a binary number, that just means you start with the lowest possible value and count up, testing each successive value. Such a brute-force method will take time, even with a computer doing the guessing/testing. And

today, it is possible to have a large number of computers collaborating on doing the guessing. With a 128 bit key, and a million computers each making a million guesses per second, it would still require 5.4 x 10^{18} years to test every possible value. To think of it another way, imagine a locked briefcase with a three-digit lock (fig. 6–27) and how long it would take to try all 1000 possible combinations (from 000 to 999) if you could try one each second: just 16 minutes. Now imagine if the briefcase had an eight-digit lock. Trying all the combinations from 00,000,000 to 99,999,999 would take you a much longer time (11.5 days if you worked non-stop), and each added digit would increase the time required ten-fold. Long keys are considered as being *cryptographically strong* due to the time and effort required to break them.

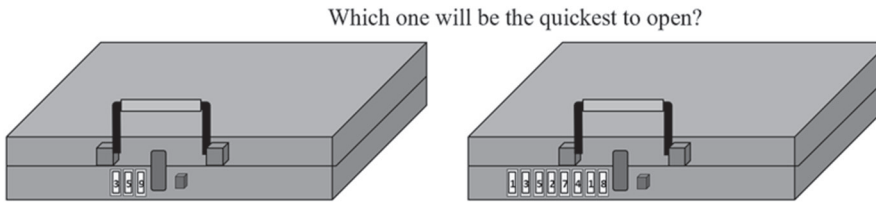


Fig. 6–27. Key size increases time required to break

Realistically, we still do not want to keep using the same key value for very long periods of time as this increases the odds of the key being broken. So, we usually limit the usage time and then require a new key. The problem is in how to distribute a new key to everyone who needs one without risking the new key being captured/exposed. And, if too many people have the key, then it is ever more likely to be revealed. (A famous quote says that "three people can keep a secret as long as two of them are dead"). Fortunately, two mathematicians devised a scheme and algorithm that can be used to generate a pair of keys that have an interesting property: if you encrypt with one of these keys, you can only decrypt with the other key. This form of asymmetric cipher is called *public key encryption*.

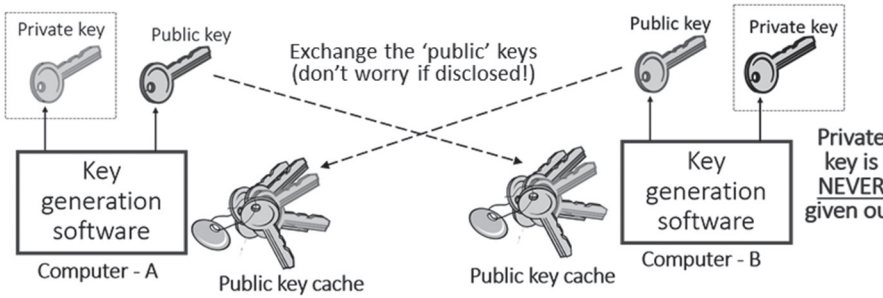


Fig. 6–28. Public and private key generation and exchange

This generated-when-needed pair of keys enables us to create an interesting encryption scheme whereby one key is kept secret and never revealed (called the *private key*) and the other key is given to anyone and everyone with whom you need to communicate (called the *public key*). When one computer needs to send a message to another, they first exchange public keys and then perform a double encryption with those keys plus their private keys. Figure 6–29 shows the general idea: Fred wants to send a message to Mary and wants to keep it confidential. So, Fred uses his private key to encrypt the message and then encrypts that again with Mary's public key. When Mary receives the message, she first decrypts with her private key (which only she has, so no one else could perform that step—so the message is kept secret) and then decrypts again with Fred's public key (which proves, if it works, that Fred's private key was used, and only Fred has that key—so Fred sent the message). This scheme ensures that messages remain confidential and that you can authenticate the sender of the message. This scheme also means that you generate new public/private keys when needed and exchange public keys with whomever you currently need to communicate with (without worrying that the public key gets disclosed) in a secure manner. The public-key, private-key encryption scheme is used in a wide range of cybersecurity applications, including secure Web browsing (https), email exchange and in setting up VPNs. This is asymmetric encryption since different keys are used for encryption and decryption. There are additional cryptographic mechanisms that can be used to verify that a message has not been altered in transmission (or to detect that it has been).

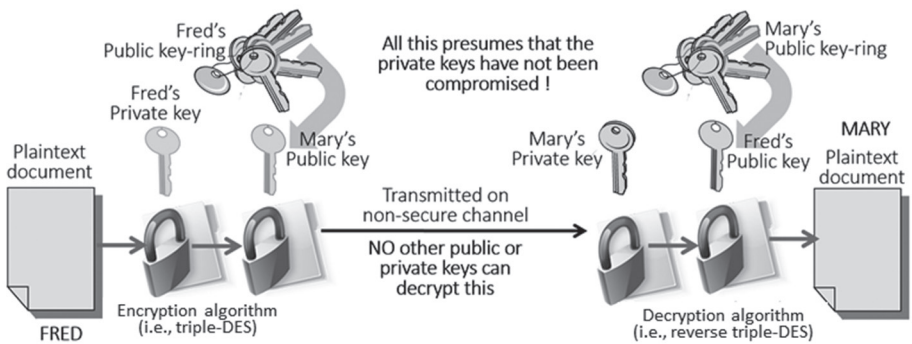


Fig. 6–29. Public-private key encryption

There are similar approaches taken when protecting information stored on a computer (a.k.a. information "at rest"), as compared with protecting the confidentiality of messages sent across a communication channel. All information stored on a computer will be in the form of files that are managed and controlled by the file manager services of the operating system. All such file managers have basic protective mechanisms that are intended to keep a user, or application, that isn't authorized to access a file from doing so. File access controls usually encompass the right

to read, copy, alter, delete, and execute. (Execute is for the case that the file contains an executable program.) Denying execute rights to all except for designated individuals is one way to prevent unauthorized use of SCADA system utility programs. Any given user typically has all of these rights over their own files. A user will have some classes of system programs for which they are granted execute authority. A given user will *not* normally have any of these rights over the files of another user, unless that user allows those rights or the two users are in a common group and the file protections are extended to the group. File protection mechanisms (like cheap locks) are mainly intended to keep the honest people honest. Hackers have found numerous ways of bypassing such protective mechanisms.

Information stored in files within a computer can be encrypted in all modern operating systems (Windows variations and Linux variations). The reason for doing this is to ensure that if an adversary were to gain access to your system (or remove a hard drive from your computer), any files they extracted would be useless to them. If the attacker's goal was to alter critical SCADA configuration/database files, the encryption would prevent that as well. They could potentially corrupt or just delete those files, but not edit/alter them in an undetectable manner. Windows has supported its encrypted file system (EFS) since the introduction of version 3.0 of the new technology file system (NTFS). EFS allows the encryption of individual files, individual directories (all files in the directory including any sub-directories), and of the entire drive (all files on the drive). EFS is user-transparent: the user isn't aware of the process of file encryption and decryption. Individual files are encrypted/decrypted with a single file encryption key (FEK), which is then itself encrypted and stored in the alternate data stream (ADS) associated with the file. FEK key encryption is done with a public/private key pair generated by the EFS for the specific user account (the user that "owns" the file). Many applications, such as Microsoft's Office applications, allow you to encrypt your work files with a user-selected password. This is not user-transparent since you have to provide the password and request file encryption, and it is up to you to remember the password you entered or your files are unusable. An EFS takes care of all of that and even allows you to encrypt files placed onto removable media, as well as on your hard drives, as long as the media uses the NTFS file structure (fig. 6–30). EFS is fast enough that most applications never notice the time required for file encryption and decryption. The details for file encryption mechanisms used on Linux systems are generally the same, although there are several available file encryption modules such as DM-Crypt, LUKS, and eCryptfs. Not all of them allow encryption on individual files, but all support full, user-transparent drive encryption.

Aside from EFS, Windows also supports what they call *BitLocker*, a full volume encryption mechanism, beginning with the Windows Vista release. BitLocker uses AES encryption and either a 128 or 256 bit key. When BitLocker is first enabled, the provisioning process (encrypting all the files) can take several hours, so it's a bit messy to use on an already running system. Just encrypting a new flash drive can take nearly half an hour. You have the option of setting a PIN that will be required whenever you boot up the computer, as an added security measure. BitLocker also

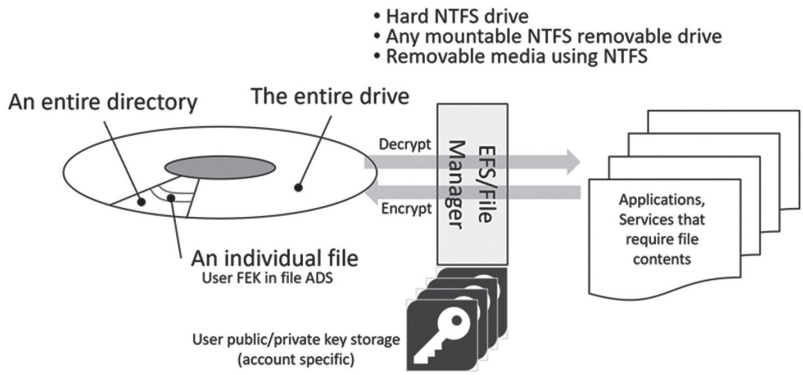


Fig. 6–30. MS Windows Encrypted File System

requires that your computer have a trusted platform module (TPM—a.k.a. ISO/IEC 11889—which is a microcontroller cryptoprocessor that performs cryptographic functions), which runs an authentication check on your hardware and firmware at startup and which will prevent a normal full-boot if it detects unauthorized changes. If you don't have a TPM chip in your computer, there is a Group Policy Editor work-around. BitLocker is quite useful for protecting portable laptop PCs that may be subject to theft or clandestine compromise.

Hash code

Although it would seem obvious that an encryption algorithm needs to be reversible to be of any use, there is actually a special category of encryption algorithm called *one-way algorithms.* These algorithms generate a unique, fixed-length, binary number (rather than ciphertext) for a given set of input data—that is, a number that would change, possibly quite significantly, if any part of the input data were to be altered. (These are called one-way algorithms or *hash algorithms* because the original input cannot be reproduced given the hash code generated, even knowing the algorithm used. It would be like trying to unscramble an egg.) When data are sent across a communication channel, whether encrypted or not, there is always the possibility that the message could be scrambled/damaged by noise, interference, or equipment problems aside from any intentional tampering.

To determine whether messages have arrived unscathed (unaltered), it is common practice to append to the messages a cyclic redundancy check (CRC) code, which is a unique number computed from the initial message. The receiver recomputes this CRC code number from the received message to see if it matches the received CRC code number. If there is no match, then there was a transmission error, and the message arrived altered. (Every RTU communications and computer networking protocol [including IP] uses a similar mechanism.) In the same manner, a data file, an executable program file, or any other collection of unchanging

binary numbers stored in a computer can be verified as unaltered or authentic using hash codes. Hash codes (a.k.a. a *message digest*) are used with large amounts of data (e.g., files)—amounts too great to validate with conventional CRC check codes, without breaking them into smaller pieces. There are many hash code algorithms, and they produce hash code values in sizes ranging from 128 bits (MD5) or 160 bits (SHA-1) up to 512 bits (SHA-512). A large message will usually be broken into many smaller pieces with each transmitted separately and each piece will have a CRC code attached. But none of those codes address the entire message, just the piece. So a hash code provides a check for the entire message.

It is common today for vendors to provide various types of files (e.g., documentation, user's guides, firmware revisions) on their Web sites for customer downloading. Usually, in addition to the files themselves, they will also provide the hash code value for each file, so that the customer can perform a hash code check on the downloaded file to verify its integrity and authenticity by matching with the published hash code value. Hash codes can be used in communications as well (fig. 6–31). We have already discussed public key encryption. Often, the message being sent will also be hashed, and the hash code value will be included in the message prior to the double encryption. This allows the receiver to verify the integrity of the message. Hash codes can also be used for identification purposes (as a *digital signature*) when combined with public and private keys. Hash codes are an essential component of cybersecurity, and we will see them applied in several cybersecurity technologies (e.g., HIDS) in Chapter 11. When you make backups of critical SCADA system files and of the operating system configurations, it is smart to run them through a hash code calculation and record those has values in a safe place. Then, if there is ever a question about the integrity of a backup, you can rerun the hash calculation and verify that the files are legitimate.

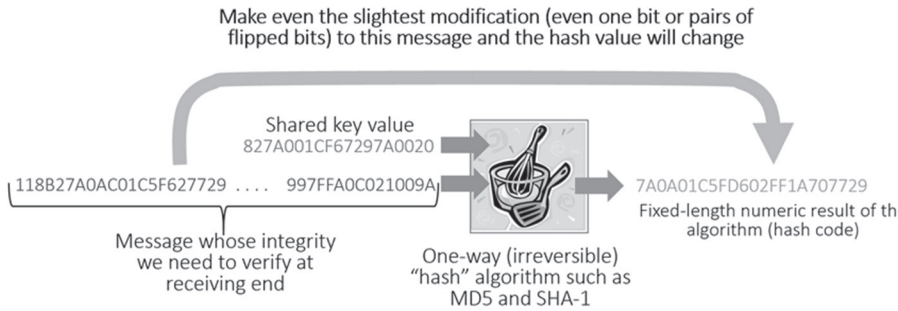


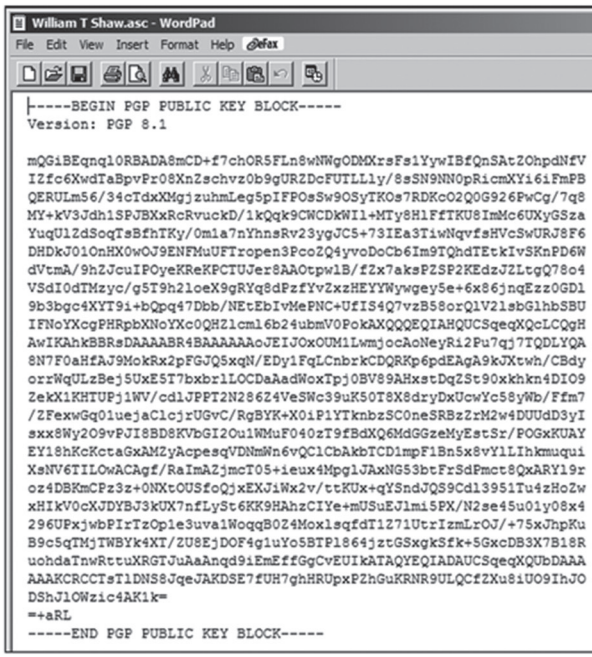
Fig. 6–31. Hash algorithm generating a message digest value

Public keys and digital certificates

We have discussed that a commonly used form of communication encryption is based on a key pair generated by some algorithm and that one key must be

kept private. When a secure communication channel is being set up—for example, as part of a VPN—the two endpoint computers/routers/gateways generate and exchange public keys (fig. 6–32) and then use the double encryption we have already discussed to negotiate a single, much smaller, symmetric key (called a *session key*), which will then be used until the VPN is terminated or some time interval (e.g., 24 hours) has elapsed (at which point a new session key will be negotiated). Public and private keys are really big, and using them for encryption takes a lot of computing power, so they are only used for the initial setup. Many applications used for communications over the Internet (such as your IE Web browser) have the ability to generate public/private key pairs as needed.

Example Public Key – Base 64 Encoding



Base64 is a binary to text encoding scheme. Each Base64 character represents exactly 6 bits of data. It looks like 8-bit ASCII but it is not. The encoding scheme allows very big binary numbers to be represented by a string of printable characters. A PGP public key can be 4096 bits long (682 Base64 characters)

Fig. 6–32. Example of a public key (Base 64 encoded)

In our discussion of VPN technology, particularly mobile/temporary VPNs, we noted the requirement to be able to authenticate the mobile computer and not just the user of the computer and that a *digital certificate* (a.k.a., a "cert") would be used for that purpose. A digital certificate is like a photo ID for a computer; it is issued by a *certificate authority* and then can be provided, when needed, as proof of identify (of the computer). Figure 6–33 shows the structure and content of a digital certificate. Currently, the standard that defines these data structures is ITU-T X.509 rev 3, and data elements have been added since the initial version. The cert

includes a public key for the user of the cert (the "Subject"), information about the issuer, and a digital signature for the issuer. A corporate IT group can get the necessary software and can be their own certificate authority and issue certificates for their own computers (particularly for any portable/mobile computers), or you can purchase certificates from any number of online sources. Certificates have an expiration date and time, and purchased ones generally last a year. But if you issue your own certs, you can make them expire in a much shorter time period.

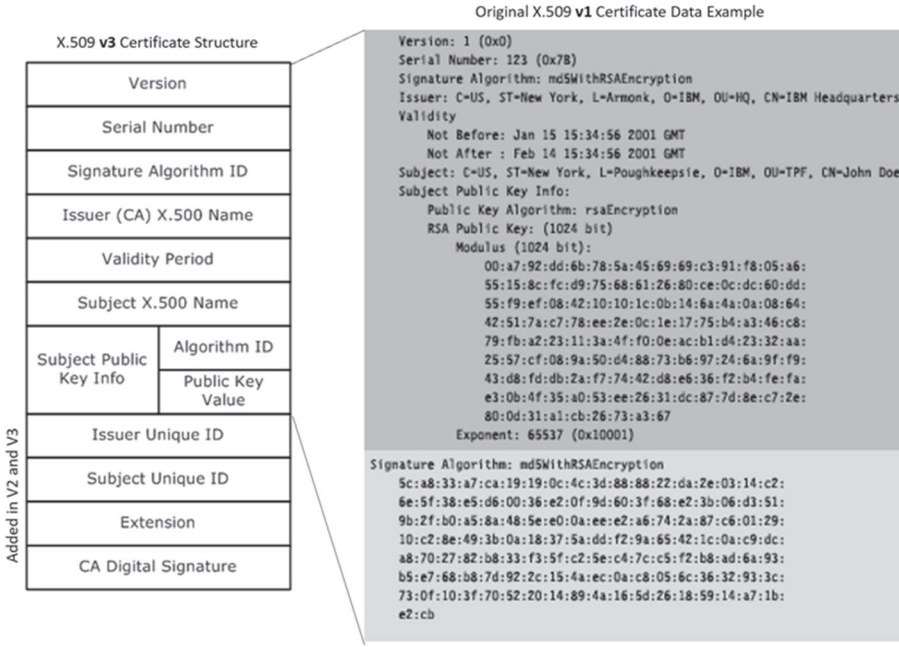


Fig. 6–33. Example of an X.509 digital certificate

A recommended cybersecurity practice is to be your own certificate authority and issue certs to any computer that will have remote access into your systems—and use those certificates for the VPN authentication. If a vendor needs to have a temporary VPN connection into your SCADA system/network, you can issue a cert with a very short duration (validity interval) and thus limit their access time, and once the certificate expires, it will no longer be accepted for continued or future use. If a company laptop is stolen, you can "revoke" the certificate and prevent the thief from being able to make a VPN connection into your network.

Antivirus protections

Another basic IT technology that is used for cybersecurity is anti-*malware* protective and detective technologies. In days gone by, you would have an antivirus

(AV) scanning program on your computer and run it periodically to detect (and hopefully remove) identified malware. AV scanning technology has advanced and changed in the past decade, but so, unfortunately, has the sophistication of the malware. We use the more general term "malware" (a contraction of the term: malicious software) rather than virus because the term virus is actually a specific sub-category of malware. There are still a number of vendors for anti-malware tools/applications, but the approach today is to try and detect malware on its way into your system, whether via a network connection or on portable media, and prevent its delivery, and not just to periodically scan memory and the file system (although that still is done). The term *endpoint protection* is used for this strategy, and this kind of technology attempts to defend and not just detect. Endpoint protection applications are a generally a mix of personal firewall, AV scanner, and HIDS technologies. A major differentiation among various AV products is how well they work in an isolated environment (no connectivity to the Internet). Testing of AV products in both online and offline mode has shown that some products heavily depend on having real-time access to the vendor's servers (a.k.a., the cloud) in order to perform adequate detection—and in offline mode, they had a poor detection performance (e.g., 95 percent on-line and only 73 percent when offline). Since a SCADA system is normally not going to have Internet connectivity, it is critical that any AV products selected, as part of an overall cybersecurity strategy, will need to have acceptable offline detection capabilities (which also means that you will have to update their signature files through some manual process and procedure rather than getting updates automagically via the Internet).

Malware in your system/computer memory or hard drive (or in a message packet or on portable media) can be identified in a couple of different ways, the most common of which is using *signatures*. Signature detection is where the AV vendor identifies a specific set of instructions in the malware (or message sequence) that are unique to that malware (and also typically computes the hash of the file containing the malware) and then looks through system files (or messages) for those same instructions (or file hashes). Signature-based detection fails when malware is new and has never been seen, because no signature yet exists. Such *zero-day* (a.k.a., 0day) malware can get past signature-based scanning. Even when it works, signature-based detection will produce a few *false positives* (incorrectly identifies something as malware) as well as generating *false negatives* (fails to detect malware that is present). With signature-based detection, you also need to regularly update your signature definitions for the latest known malware—generally every month, if not more often. This process is much messier if you need to perform it manually because of no Internet connectivity. Unfortunately, much of the more sophisticated malware today is *polymorphic*, meaning that as it copies itself and spreads to more computers, it actually makes changes to itself to help the new copies avoid detection. For that reason, signature-based detection is losing ground, and another means of malware detection is required. Currently, there are two options: *heuristic scanning* and behavior-based (a.k.a., anomaly) detection. Most antivirus programs that utilize heuristic analysis perform this function by

executing the program instructions of a questionable program or script within a specialized virtual machine (a.k.a., a *sandbox*), thereby allowing the antivirus program to internally simulate what would happen if the suspicious code were to be executed, while keeping the suspicious code isolated from the real machine and operating system. If the code appears to perform malware-like actions (reading sensitive files, attempting to initiate an ftp action, trying to change system settings, etc.), then it is flagged as potential malware. Static analysis can also be performed which can include looking at the list of DLLs the executable references in the file header. Heuristic scanning is subject to a greater level of false positives/negatives but is generally effective against 0day malware that signature detection can't address. Behavior-based detection essentially looks at what a program is doing and compares its actions against a baseline of recorded actions to see what is different and checks to see if any of the differences are on a list of known malicious behaviors (e.g., it is making an outgoing TCP connection that it never did previously). Behavior-based detection is subject to a greater level of false positives/negatives but is generally also effective against 0day malware. Behavior-based detection cannot be used for detection on portable media or in communications, because it needs the questionable program to be actually executing so it can be observed (the malware must be allowed to get into your system, or at least a virtual environment, and run). Most, if not all, SCADA vendors support the use of some form of AV scanning and endpoint protection on your computers, workstations, and PCs. Because of the computing power required for heuristic analysis and even for signature analysis, some organizations have elected to set up specialized scanning stations (a.k.a., scanning kiosks) that can be used to scan portable media, and which can have an Internet connection (through a firewall) for receiving updates. In such an arrangement, you can also use multiple AV scanning engines to enhance your malware detection likelihood. We will continue the discussion of the challenges of portable media and computer devices as attack vectors, later in this chapter.

Architectural strategies

Another way of enhancing your SCADA system cybersecurity is by using architectural strategies that can help to limit the spread of malware and block/slow an attacker that gets a foothold in your network or systems. An important concept in cybersecurity, which was borrowed from the military, is something called *defense-in-depth* (DiD). This is a strategy that says you need to place as many barriers as you can between important assets and the attacker in order to delay them, detect them, and give yourself an opportunity to respond and recover. In Chapter 11, we will discuss technical countermeasures that can be used to detect, defend, and respond to cyberattacks. But a good starting point for DiD is to use internal firewalls (and possibly also apply VLAN technology) to break up your networks into segments and to restrict communications between segments to just the minimally required traffic. In figure 6–34, we have taken the simplified SCADA system block diagram we examined in Chapter 1, and we have added firewalls to segregate the overall system

and create what the ISA's SP-99 cybersecurity working group calls "zones and conduits." A zone is a network segment containing systems/computers/devices that need to constantly intercommunicate and only infrequently communicate to other systems/computers/devices in other zones. Conduits are interconnections between zones, created by internal firewalls that are configured with a rule set that explicitly limits and controls permitted inter-zone traffic. On the diagram in figure 6–34, we have now placed firewalls at several locations so as to establish separate zones (and this includes firewalls at each IP-connected field site, which might be specialized *industrial firewalls*). We would also want to use firewalls to protect the primary and backup SCADA servers from infecting each other were one or the other compromised. Same between primary operating locations and backup operating locations if there are any. We have also created a DMZ in which we have placed a historian, which will now serve as the mechanism for providing SCADA data to corporate applications. Also, every remote connection via a WAN to remote systems and users is now secured with VPN technology, using digital certificates issued by the SCADA system owner and one or more VPN gateways. You can also assume that there would be additional detection placed on the overall network in appropriate places. We will discuss HIDS, NIDS, and SIEM technologies in Chapter 11. These changes would make it far less likely that an attack would succeed or at least would go undetected.

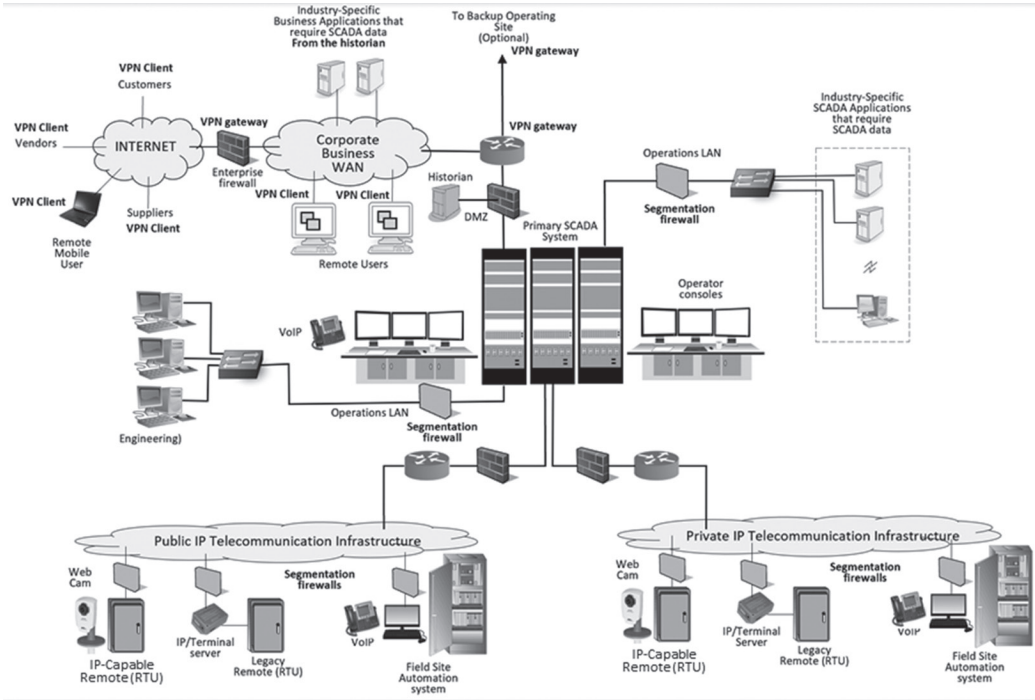


Fig. 6–34. Creating zones and network segmentation with firewalls

Portable media and portable devices

As has been mentioned, one of the attack pathways into a system is by means of computer-readable portable media. There are numerous examples of personnel unwittingly bringing malware into work on floppy disks, CDs and (these days)USB flash drives and infecting systems. The malware may have gotten onto the media because the person's home computer was infected due to sloppy emailing and Web browsing practices, and they use the portable media both at home and at work. In one documented case, at a trade show, USB flash drives were given out as booth gifts and, unknown to the vendor, it turned out that every one of the flash drives contained malware. Vendors have also been known to provide CDs with their software plus some secret malware (a well-known printer manufacturer's driver software included a bit of malware that infected every computer into which the printer driver was installed). So, portable media needs to be considered as potentially dangerous until proven otherwise. We have already discussed AV scanning and detection and some of the limitations and issues with current AV technologies. But it is still a good idea to establish procedures and processes for dealing with any portable media that is to be used for support of the SCADA system and any associated/interconnected systems. In some industrial segments (e.g., domestic nuclear power generation), a common practice is to establish a set of trusted

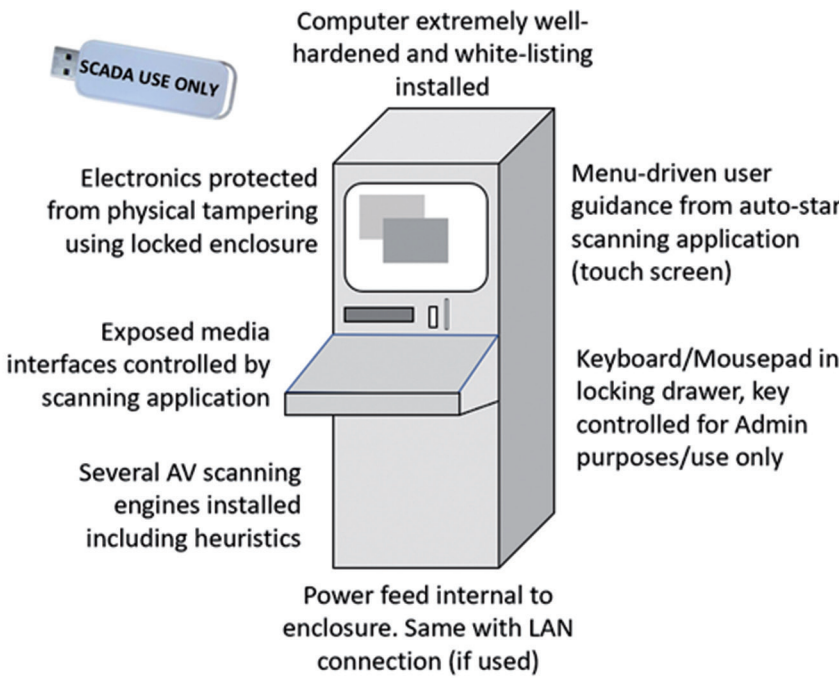


Fig. 6–35. Conceptual design of a media scanning 'kiosk'

portable media, with color-coding and labeling to identify it as such, and only permit that portable media to be used on critical systems. You could even have portable media that is computer/system-specific. The media is maintained as trusted by regularly submitting it for AV scanning, issuing it for use as needed, and physically securing it when not being used. Some corporations have purchased or constructed media scanning stations/kiosks (fig. 6–35) that incorporate several AV scanning engines, including at least one capable of performing heuristic scans—and they require that portable media to be used on any critical system be scanned prior to use. Such scanning kiosks need to be protected against compromise since they could then spread malware on all portable media they scan. It is typical to severely harden such computers and to install whitelisting to block their being infected. Companies such as Tresys (https://www.tresys.com/) and OPSWAT (https://www.opswat.com) sell media-scanning kiosk products. Some of these products can also scan hard drives if removed from a laptop or PC. Portable media can be categorized into two broad classes: media that is potentially capable of hiding its contents (sometimes called active media) and media that cannot (sometimes called passive media). A CD/DVD platter would be an example of the latter category whereas a USB flash drive with file encryption would fall into the former category. Some organizations require files on untrusted active media be copied to passive media prior to being AV scanned, and then if 'clean', those files are copied onto trusted media for use on critical systems.

When external, untrusted portable media is required (e.g., to install updated signatures, patches, or vendor firmware updates), it can also be put through the scanning process and its contents copied onto trusted media, prior to being used on any critical system.

For portable digital devices (e.g., a laptop PC), which may need to be temporarily connected to a critical network or system, a similar process might be required (especially if it is a portable device not under the control of the SCADA system owner): extract the hard drive and use a USB adapter so that the drive can be plugged into the scanning kiosk and given a thorough scan prior to allowing it to be connected. This is only required for digital devices that could potentially be used to deliver malware or to launch an attack on the connected system—so things like digital multimeters, digital oscilloscopes, hand-held instrument calibrators, and other specialized test equipment would not need this scanning.

In Chapter 11, we will go into much greater detail about countermeasures that can be used to establish adequate cybersecurity for your SCADA system, but in this chapter, we have tried to establish a base of terminology and concepts that will make the discussions in Chapter 11 much more comprehensible.

7

Identifying cybersecurity vulnerabilities

Threats and Threat Agents

When discussing the possibility of a hacker, a malware infection, or a terrorist attack on a SCADA system, we are really referring to the probability (or likelihood) of an attack occurring and the consequences resulting were it to be successful. An attacker, regardless of type and motivation, will be looking for the weaknesses in your defenses and will attempt to exploit those weaknesses (vulnerabilities) to carry out an attack. It is important to take reasonable actions to protect your systems, particularly if a successful attack upon them could cause loss of life, injury, or substantial damage, not to mention any additional financial impact. But what is the probability of an attack and the possibility that it could succeed, and who are the potential attackers?

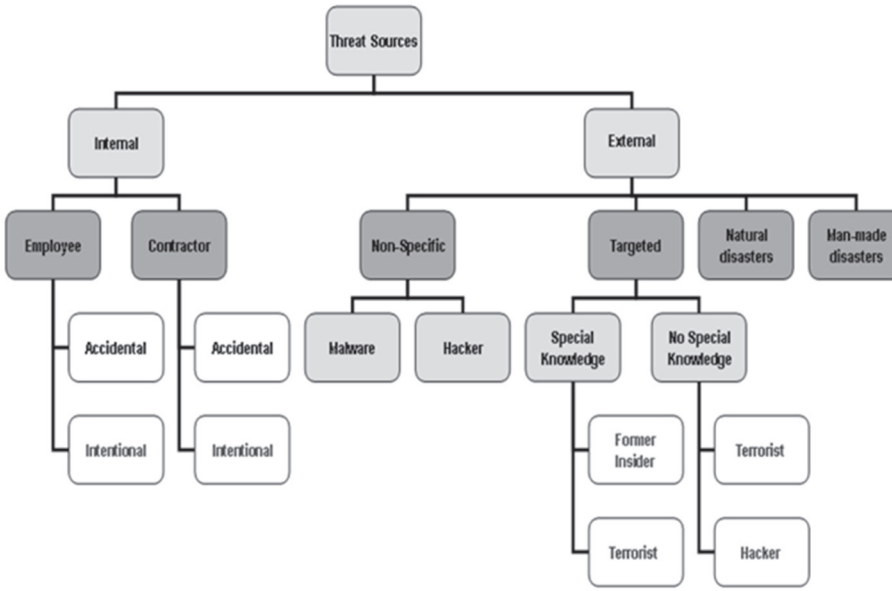


Fig. 7–1. Taxonomy of potential threat sources to a SCADA system

The U.S. government (particularly the National Institute of Science and Technology [NIST]) has done a lot of work in the post-9/11 environment, classifying and categorizing cyberthreats. They use the term *threat agent* to describe the potential source of a threat. Everyone thinks cyberthreats are about clandestine cells of anti-U.S. hackers attempting to bring down our government or industry by launching virus attacks across the Internet. In truth, however, the threat agent that knocks down a critical SCADA system could be a tornado, an earthquake, or a member of the cleaning staff who spills a bucket of water on a server. When anticipating the possible threat sources, we need to take a broad view. A good starting point for sorting out potential threat agents is to separate them into two basic categories: internal threats and external threats (fig. 7–1).

Internal threats

Internal threats (threat agents) are people currently working as employees of, or as on-site contractors to, your organization, who therefore have greater physical access rights than someone who doesn't and probably have special knowledge not available to an outsider. That certainly means physical access to a corporate facility, but possibly even includes physical access to the SCADA system equipment or auxiliary subsystems (e.g., UPS/power/telecom/LAN) that support the SCADA system. That person may also have greater access to confidential and critical information (e.g., passwords or IP addresses) and even some level of electronic access rights (e.g., an account on the corporate computer system). An internal threat can be an actual, direct-hire employee or a support contractor, such as someone from among the janitorial staff, maintenance personnel, or even a vendor's technical service personnel (part of the *supply chain threat*).

Internal threat agents can pose intentional threats or unintentional/accidental threats. Accidents and errors are dangerous because of the physical and electronic access they have. Intentional threats are even more dangerous, because of their specialized access and knowledge. The most acute level of threat would be posed by a high-authority-level, internal threat agent who intentionally wishes to inflict damage—regardless of whether that person is a direct employee or a contractor. As it turns out, disgruntled current and former employees can sometimes be the most dangerous threat agents. Google search the name Vitek Boden, and you will see an example of a former contractor who used his special knowledge to commit sabotage on a SCADA-DCS installation in Australia. To date, there have been several instances of insiders taking malicious actions to disrupt the operation of industrial automation systems, so the possibility can't be totally discounted. In the domestic nuclear power industry segment, they have something called an insider mitigation program (IMP), whose objective is to identify potential malicious insiders; the program uses initial and periodic background, legal, and financial checks of personnel with access to critical assets. The objective is to identify individuals that may be targeted by threat agents and compromised because of debt, drug use, gambling

problems, etc. It also teaches personnel to watch for signs of abnormal or suspicious activity on the part of others.

External threats

In the simplest possible terms, external threats are everything else (that which is not an internal threat agent), including natural disasters (e.g., flood or earthquake) and human-made disasters (e.g., riot or war). The main difference is that an external threat agent does not have authorized physical access rights to your facility and critical systems and would not (or should not) have access to your confidential and sensitive information.

A critical subcategory of the external threat agent is the *former insider*. This could be an ex-employee who quit just last week and still has confidential information because no one has gotten around to changing the corporate access-control list or deleting his account ID and password from the computer systems. A former insider will also know a lot of things that come from having worked in your organization, and such experience can't be taken away as easily as a key to the building. A former contractor (or security consultant) would fall into this category as well.

Even threat agents that have never been insiders can pose a serious threat. It is a mistake to believe that a determined attacker can't obtain an amazing amount of confidential information, if they are willing to spend the time and money. Keep in mind that the same SCADA vendor that supplied your system will sell an equivalent system to anybody with cash, and anyone can sign up and attend their training courses. Many vendors even provide access to their user's manuals and documentation on their Web sites.

A serious attacker can also find out information by locating and bribing or threatening ex-employees of the SCADA vendor and of your own organization. Also, the basic operating systems and LAN/WAN technology of most SCADA systems today are going to be either a Linux variation or a Microsoft Windows offering (or a combination of both), both of which are well known and profusely well documented. If a terrorist organization decided to target your systems, they would probably go to great lengths to get someone into your organization, even if just as a member of the janitorial staff. It is amazing to consider that the most secure and restricted spaces in an organization still need to be cleaned and that the cleaning staff often have unlimited and unsupervised access to those spaces. Today, we also recognize the need to address the 'supply chain' as a threat source. Counterfeit materials and software have been sold to corporations and then discovered to contain malware or backdoors or trojan horse functionality. One electric utility found out that a shipment of 20 laptop PCs from a major Asian manufacturer contained counterfeit versions of Windows with data exfiltration malware preloaded. (Fortunately this was detected due to supply chain testing procedures.) It is not out of the question that an adversary might even attempt to

gain employment with one of your major vendors (particularly in field service or product support) in order to take advantage of the trust-relationship between your organization and that of the vendor.

Targeted attacks

Another broad categorization of threat agent differentiates between those that are targeted and those that are nonspecific. If a hurricane or a flood hits your facilities, it is probably just pure bad luck and not because Mother Nature has specifically targeted you. The same is probably true if a virus or a worm gets into your systems through some unprotected network connection or infected portable media. If a hacker discovers a pathway into your systems or network by war driving, it is probably also just pure bad luck. (Still, shame on you for leaving your systems exposed!) The most serious threat comes from an attacker who is specifically targeting you and your systems. That attacker could be a terrorist organization with significant resources or just a disgruntled ex-employee who is a part-time hacker—or even an extremist group (political, religious, nationalist) that feels it has the right to teach you a lesson lesson and punish you for going against their strongly held beliefs. Depending on your specific industry, there could also be a financial incentive to attack your systems.

The targeted attack is one that will generally be well planned and executed, and it should be assumed that the attacker will do everything possible to obtain critical confidential information and access to your systems, through whatever means are necessary. You should assume that they will have the financial resources to hire the technical talent they might need. This is a good point to introduce another security term and concept: *social engineering*.

Social engineering

Social engineering is the process of obtaining unauthorized confidential information and possibly obtaining unauthorized physical access, through the manipulation of people. Con artists have been using these techniques forever. Social engineering relies on learned behaviors, habits, manners, and basic human nature (good and bad). It also often involves exploiting the nature and culture of organizations and specific individuals.

A prime example of a somewhat obsolete social engineering strategy for obtaining confidential information about your organization would be a technique called 'dumpster diving'. It is amazing what people will toss into the trash (at least, they used to do so), on the assumption that this act alone makes it disappear. (Table 7–1 outlines some well-known social engineering techniques.) This notion isn't limited to paper trash either. Corporate information and secrets have been obtained by pulling the disk drives out of discarded PCs and reading the contents of discarded floppy disks, CD/DVDs, and other removable media. A good procedure for the physical destruction of discarded printed and electronic media—including

electronic equipment (computers, tablets, cell phones, etc.)—is essential to ensure that these materials don't fall into the hands of a threat agent/attacker. Sensitive documents and CD/DVDs containing sensitive information should be shredded. You should actually have an *information protection policy* that specifies what types of information, regardless of form and format, require some level of protection and destruction. Most people would not toss a detailed system/network drawing with IP addresses and equipment details in the trash. But simple, innocuous documents, like an internal phone-number list and the corporate password policy, can also give an attacker information that can aid in staging a cyberattack. If computer equipment is declared as obsolete, all hard drives should either be removed and damaged or put through a multi-pass "erasure" procedure (make the contents unrecoverable by writing several passes of random values over the entire drive). Conventional magnetic media disk drives can also be erased with powerful magnetic fields (a.k.a., degaussing), but this does not work for solid state disks. If full-drive encryption is applied to a hard drive, then erasure is usually not required because the contents are unrecoverable without the encryption keys (which should be deleted). A related problem occurs with unprotected portable computers, tablets, laptop PCs, and cell phones that are stolen (which is a definite social engineering technique). They can also contain sensitive information that would then be disclosed unless protected in some manner. (They may even be configured for remote VPN access into your corporate network!) This is another reason to consider enabling the EFS capabilities (and BitLocker for Windows) on these devices, in addition to having restricted-user accounts with strong passwords. The following table (7-1) lists examples of social engineering tricks:

Table 7–1. Example social engineering techniques used by attackers

Technique	General Description
Dumpster Diving	Obtaining information by going through the trash in a facility's dumpster. Today more likely to be going online to try to buy used computers being sold off as obsolete.
Tailgating	Following personnel in through locked doors once they use their key/badge to unlock the door.
Role Playing	Calling help desk and acting like an angry executive who has been locked out of computer and needs a new ID and password assigned right now or dire consequences.
Phishing/Spear Phishing	Sending carefully crafted emails with convincing details to get recipient to open malicious attachment or click on a malicious hyperlink.
Help Desk	Calling personnel, claiming to be help desk returning their call, until you find someone who wants help and then getting their user ID and password so you can "help" them.
Flash Fishing	Leaving infected media in public place (lobby) and labeling it "Executive Salaries" or similar to encourage the person finding it to insert into their PC to examine contents.

Social engineering also involves using other tricks to get people to divulge confidential information. Calling around through an organization and using what you learn from one person to leverage your conversation with the next person is another common trick. ("Hi Bob, Joe over in accounting told me to call you when Larry and I saw him at lunch—you know, Larry from tech support? Right, well, Joe said that you could tell me. . ."). Calling a company IT help desk and faking an emergency, to get a new ID/password issued, is a commonly used trick (especially if you sound irate and imply that they will take the blame for what will happen if you don't get that new ID right away).

Gaining entrance to restricted, access-controlled areas is often achieved through a number of tricks. Bringing a pizza or flowers and claiming they are to be delivered to person X up in the control room or computer room can often get an invader into those areas. Timing arrival at a secured door, with hands piled high with stuff, so as to reach the secured door at the same time as a legitimate employee, almost always guarantees that the employee will open and help the invader through the door. Fumbling with a key card or access token (a fake) or asking the employee to reach into your pocket to get your key card further increases the success rate of invasion.

Social engineering can also be used to get malware into otherwise isolated systems. For instance, a clever hacker once made a CD containing a serious virus, then made a fake music label for the CD and placed it in the jewel case of the real music CD. One morning, he left this CD by the entrance to the facility of the company whose systems he wanted to penetrate. As expected, someone picked up the CD, thought themselves lucky, took the CD to their office and inserted it into their PC, from which a computer virus subsequently spread onto the corporate network. Today, the most common means for attackers to gain a foothold in your corporate WAN is to employ a social engineering tactic called *spear phishing* or a similar one called *whaling*. Several threat intelligence organizations state that phishing attacks are the leading means by which cyber attackers compromise computer systems (particularly home computers). Attackers make use of company Web sites, company press releases, company job postings, and social networking sites, such as Facebook™ and LinkedIn™, to gather enough information to create a highly credible (and specifically targeted) email message in an effort to get the recipient to click on a hyperlink that redirects the recipient's computer to a malicious Web site that attempts to inject malware that targets known Web browser vulnerabilities (fig. 7–2).

Spear phishing is different from basic phishing in that basic phishing is usually an email sent to a huge number of recipients, many of whom will immediately realize the email is fraudulent, because it will contain elements they recognize as inaccurate or non-applicable (e.g., a message from a bank about your account, but it is a bank with which you have no account). But the attacker's hope is that at least a small percentage of recipients will click the link and get infected. If the target of the phishing effort is a corporate executive or senior manager (i.e., a 'big fish') then the term whaling is used to describe the attack. Spear phishing remains one of the

most serious threats and is the most common means for gaining access to corporate computers and corporate networks. A good cyber security practice is to send phishing emails to your own personnel to remind them to remain vigilant.

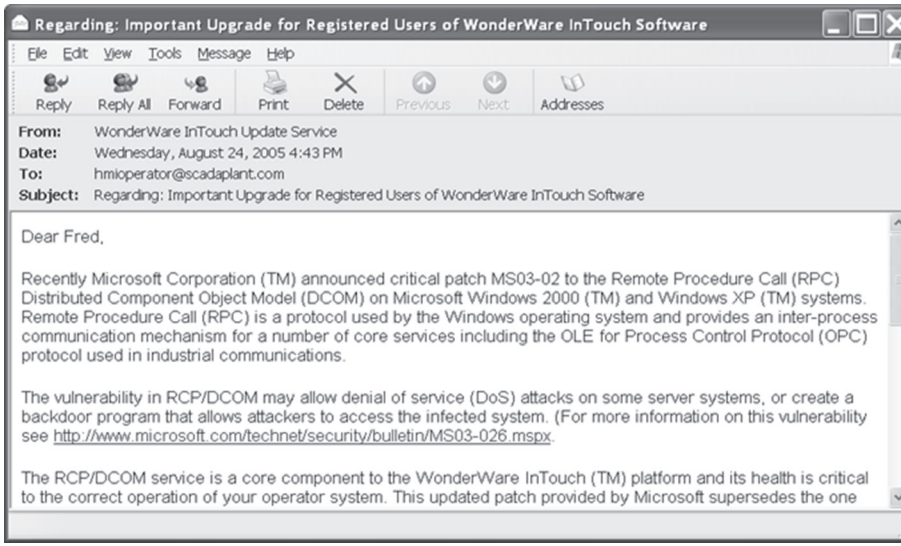


Fig. 7–2. Example of a spear phishing email

Many of us have RFID badges with our photo and other information, which act as our ID badge, but which also unlock doors when we pass them by/over a badge reader. A simple badge cloning device can be built using a small, battery-powered, inexpensive, hand-held computer called a *Raspberry PI*, and an RFID antenna and reader module. With this device, if you can get within a couple of feet of a person with an RFID badge, the scanner can read the contents just like the reader at the door. Then, back at the locked door, it can be used to replay that content, convincing the door access control system that you are the person whose badge you cloned. In the Federal government, to prevent this, all ID badges are required to be placed into special holders that block the signal (they form a *Faraday cage* around the badge). This is also why scan-proof wallets have gained in popularity. Of course, a less technical social engineering ploy, which also gets you into that locked door, is to follow company personnel when they go to lunch and steal one of their badges from off a jacket or coat. This is why adding a keypad and requiring a PIN to be entered enhances physical security, as compared to just having an RFID badge or ID card.

The best defense against social engineering is an employee education program on cybersecurity basics, including social engineering tricks, and well-established policies and procedures that make such tricks unlikely to work. If you can deny

targeted attackers the critical information they need in order to be successful, then you can greatly reduce both the likelihood of an attack, as well as the likelihood of an attack being successful. If an employee knows that letting someone else into a secure door using their own badge is prohibited, then they are more likely to ask the person to go to the guard station or main entrance and foil the attempt.

How much is enough? There is no single, simple answer to the question of how much effort and money should be put into building and maintaining cyber defenses. If you are answerable to regulatory agencies that have defined a basic set of requirements (e.g., those established by the North American Electric Reliability Corporation [NERC] or the Nuclear Regulatory Agency [NRC]), then you are obligated to meet those requirements. Your legal staff and insurance underwriters will probably also have some thoughts on the subject, depending on the liabilities and exposure a successful cyberattack would bring.

Most critical SCADA systems have been built with a lot of backup and redundancy, even well before the events of 9/11. They are intended to survive a certain amount of damage/failure with either no degradation or at least graceful degradation (they have a *fault-tolerant* design). Nevertheless, they were probably never designed to withstand an intentional attack (unless they are rather new). SCADA system designers have always had to deal with the possibility of communication outages, and they usually build backup or fault-tolerant communications systems. SCADA systems have incorporated redundancy in their architectures since they were introduced, because computers break.

Supervisory control has never been a substitute for local controls and safety interlocks. No pipeline control system designer would (should) allow local processes to go critical just because the supervisory system couldn't communicate with the local RTU or PLC. Also, many of the processes typically monitored and controlled by SCADA systems have a lot of inherent stability or capacity (along with local controls and safety systems). In other words, just killing or disabling a SCADA system isn't particularly easy—and even if that were to happen, the results might not be catastrophic, or there might well be sufficient time to recover before a catastrophe could occur. Those factors impact the risk posed by a cyberattack and thus the level of protection (and incident response) required. The biggest threat from a cyberattack might be the usurpation and malicious exploitation and manipulation of the SCADA system, and preventing that possibility ought to be where the most effort should be focused. These points need to be considered when making the decision about how much protection is enough and what type of protection is required.

The federal government or, more specifically, the Department of Homeland Security has established a list of critical infrastructure sectors whose assets need to be protected against terrorist attacks because "their incapacitation or destruction would have a debilitating effect on security, national economic security, national public health or safety, or any combination thereof." The 16 Critical Infrastructure sectors are:

- Chemical
- Commercial Facilities
- Communications
- Critical Manufacturing
- Dams
- Defense Industrial Base
- Emergency Services
- Energy
- Financial Services
- Food and Agriculture
- Government Facilities
- Healthcare and Public Health
- Information Technology
- Nuclear Reactors, Materials, and Waste
- Transportation Systems
- Water and Wastewater Systems

SCADA technology plays a major role in several of these sectors, and thus a SCADA system is one of those digital assets that needs to be protected. In some industries, there is a challenge getting corporate or plant management to provide funding for cybersecurity programs. This is usually because they have not seen any evidence of a cyberattack on their facilities, and it is hard to do an ROI calculation because cybersecurity is basically a cost of doing business, just like paying for liability insurance. Plus, implementing a cybersecurity program is not a one-time event. There is the initial cost to put it in place and then ongoing annual expenses to maintain the program and adjust it, as needed, to deal with the changes in technology and in the *threatscape*.

A classical discussion of risk assessment and mitigation would involve a series of statistical calculations regarding the cost of replacing a critical asset, the cost of losing the use of the asset (and any related costs for damages and liabilities), and the likelihood and frequency of possible threats actually coming to pass. Those sorts of computations and decisions might be relevant when discussing the chances of losing an e-commerce Web server, but they are not as clearly relevant to considerations of SCADA security. This is because losing a SCADA system has impacts beyond just finances or business considerations; rather, it may present a public safety issue. If someone can disrupt the electricity, natural gas, or water supply by attacking a SCADA system, then a typical cost-benefit (financial) analysis may not be appropriate for justifying the investment in the necessary protective countermeasures. (There may be regulatory or legal requirements that would initiate punitive actions against organizations that could not show that they took reasonable and adequate measures to protect their SCADA systems.)

Obvious Points of Attack and Vulnerability

To reduce the probability of success of an attack, steps must be taken to eliminate potential points of vulnerability. However, what are the points at which an attacker will focus, and what types of attacks can be used? We will discuss physical and operational security later (see Chapters 9 and 10) and go into more detail on cybersecurity (see Chapter 11). For the moment, we will focus on cyberattacks, rather than physical attacks. The objective of a risk assessment is to identify vulnerabilities that can be exploited by an attacker and to implement countermeasures to either eliminate the vulnerability or reduce (mitigate) the consequences of an attack against that vulnerability (either approach is equally valid). A cyberattack requires three elements: (1) an identified exploitable vulnerability, (2) a pathway that provides access to that vulnerability, and (3) an exploit that can take advantage of that vulnerability. Unless you have all three, then you can't stage a cyberattack. (Were a computer chock-full of vulnerabilities but locked away in a fortified concrete bunker with no external communications and with no one allowed into the bunker, then all those wonderful exploitable vulnerabilities would be useless to a potential attacker.) Good cybersecurity comes from both eliminating vulnerabilities and eliminating (or defending and monitoring) attack pathways. *Countermeasures* are used for this, and they can be technical (e.g., virus scanning software), procedural (e.g., a policy that disallows wireless APs from being attached to the SCADA network/LAN), or even physical (e.g., the USB ports on the computer have a physical barrier [port blocker] installed to prevent their use). It is also important that you be realistic in your efforts. It may be outrageously expensive to implement all possible countermeasures in order to reduce cyberthreats to a zero likelihood. On the other hand, a small investment might reduce them to an acceptable level of likelihood. Addressing and balancing all of these issues is part of a risk assessment process.

Figure 7–3 shows possible pathways through which a cyberattack could be perpetrated on your SCADA system. For the sake of older, legacy systems, we have included the possibility of a telephone connection that would enable an attacker to gain remote access to the command-line interpreter of the computer. Today, that is far less likely to be an issue or even a possibility, which is why we have indicated that this pathway is more of a legacy issue. We have also shown the communication polling channels out to the field sites and RTUs, but here we are talking about serial communications, not IP networking to the field. IP networking to the field falls into the category of wired WAN connectivity. In an earlier chapter, we discussed serial protocols and the likely impact an attacker could have if they could access such a communication channel. Because of the way serial communication channels and serial SCADA/RTU protocols work, this is not a likely viable attack pathway for a compromise of the SCADA system. Not to say that an attacker could not cause some damage if they gained access to one or more of those channels—after all, the RTUs may operate equipment that, if manipulated maliciously, could inflict significant damage. The point was that this pathway would not allow the SCADA system

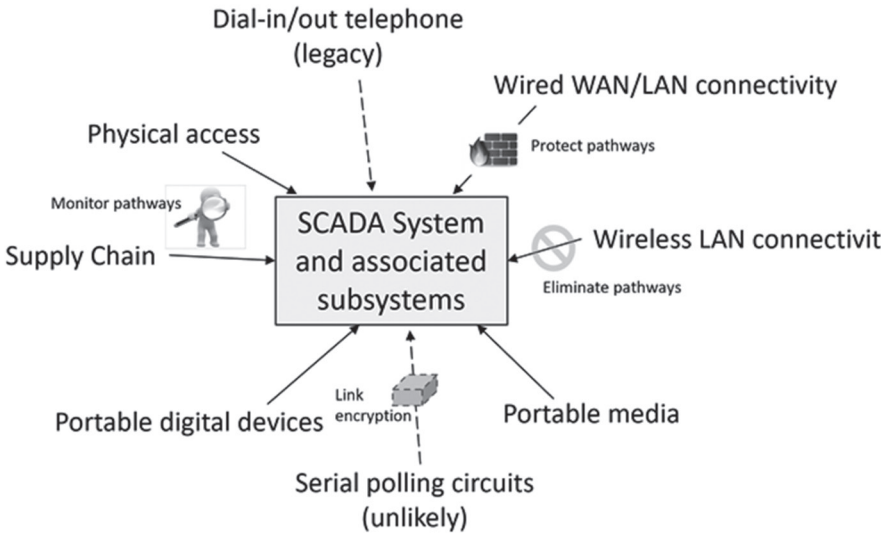


Fig. 7–3. Typical attack pathways for cyberattack

itself to be compromised (e.g., malware injected.) And, of course, serial channels to the field can be protected through the installation of serial link encryption units at each end of the channel.

For the pathways that are possible and viable, the important thing is to see if any can be eliminated (e.g., forbidding wireless functionality in all computers and network-connected devices) and, if not, whether they can be defended (e.g., placing a firewall on LAN/WAN connections) or, if that is not possible, whether there can be at least some form of detection/monitoring along the pathway to attempt to detect and then block attacks (e.g., making physical checks of received portable media to verify tamper-indicating packaging has not been disturbed; performing background checks on all personnel permitted physical access to the SCADA system; requiring vendors to submit their laptop PCs for scanning prior to use). In Chapter 11, we will discuss various cybersecurity detective and protective countermeasures in detail but let us at least further define the term.

Countermeasures

Countermeasures are actions taken or measures employed to counter a viable threat, be that threat physical or cyber or a combination of both. The countermeasures generally fall into three categories: (1) physical countermeasures (e.g., door locks), (2) technical countermeasures (e.g., network intrusion detection), or (3) administrative countermeasures (e.g., procedures, policies, and training). All three types of countermeasures can be used to achieve varying levels of physical security, operational security, and cybersecurity (fig. 7–4). Good cybersecurity generally requires/

presumes that you have at least adequate physical and operational security as well. If anyone can just come into your facility and walk into the control room and plop down in front of an operator's console and start operating field equipment, then the quality of your cybersecurity doesn't really matter much. Physical countermeasures can help you implement good cybersecurity (e.g., port blockers on unused USB and Ethernet ports), and technical countermeasures can help you implement good physical security (RFID keycard and PIN to get past the restricted area access control and alarm system). This book is focused on the cybersecurity aspects and not physical security, per se, but we will discuss physical security as well because you need it also. In fact, in the electric power generation and transmission sector, the NERC CIP standards require the establishment of an adequate physical security perimeter around the SCADA and DCS system assets.

	PHYSICAL Security	OPERATIONAL Security	CYBER Security
Technical Countermeasure	Key-Card access control system	USB whitelisting by serial No.	DMZ Firewall
Administrative Countermeasure	Background checks	Detailed Backup Procedures	Account deletion on job termination
Physical Countermeasure	Locked cabinets with controlled key distribution	Warning Signs and Lights	Server room access via guards

Fig. 7–4. Classes of security and types of countermeasures

Vulnerabilities

There have been, and continue to be, a large number of known, exploitable vulnerabilities in both the various Linux and Windows operating systems and applications (and others as well, but these are the big two). Such vulnerabilities are constantly being sought by both researchers and hackers. Once identified, the goal is to develop either a software *patch* or software update that can be applied to eliminate the vulnerability (or, if you were one of the bad guys, to create an exploit that can attack and abuse the vulnerability). Applying available patches and updates is an administrative countermeasure, and it is important to have a formal policy and applicable procedures regarding patches and updates. As was mentioned, a vulnerability is not important if an adversary can't exploit it (no pathway). When SCADA systems tended to be isolated and air-gapped, the urgency of applying updates and patches was far

less than it is today with so many possible pathways of attack. Nevertheless applying (and testing) patches and updates is a necessary evil today. It is unlikely that your SCADA vendor, who basically just sold you software, will be providing you updates and patches to address problems in software they did not provide (like your operating systems). But your corporate IT group should be keeping track of patches for all of the various software used in both the business and automation systems and should have a process and procedure for ensuring that patches get reviewed, tested, distributed, and applied (Microsoft also uses the term *hotfix* for major patches that fall below the level of a full update). Preferably, there ought to be a test bed where patches and software updates can be installed and given at least a basic functional test. There have been software updates from Microsoft (and others) that caused major malfunctions in both business and automation systems, so blindly applying them to a production SCADA system, even a redundant one, is a risky strategy. Since most SCADA systems are fully redundant at worst, if no test bed exists, you ought to be able to test patches and updates on the backup/standby components of the system. Since a SCADA system generally should not have direct connectivity to the Internet, automatic updates and patch application cannot be supported (and is not really all that desirable, due to the risks and attack pathway that would create). But Microsoft has support for WSUS (Windows server update services), which allows a computer that does have Internet access to receive updates and patches and then distribute them internally to other systems. There are similar utilities for distributing Linux patches and updates, some of which are free open-source and some commercial products such as Patch Manager Plus™. Patching to eliminate identified vulnerabilities has become a routine and highly necessary procedure. All major software vendors find bugs in their software, and many of those bugs turn out to be exploitable. In the last decade, Microsoft has disclosed more than 3,500 security flaws in its products, at the rate of several per week (and thus the need for "patch Tuesdays"). Some 50 percent of those flaws involved errors that allowed *malicious code execution*, meaning that a payload of malicious program code could be delivered and executed by exploiting that flaw. Exploits were created to take advantage of nearly 250 of those flaws. But Microsoft is not alone. In second place behind it is Oracle, with a tally of over 3,200 disclosed vulnerabilities in the last decade. Apple's products, generally perceived as being more secure than Microsoft's software, piled up over 2,600 vulnerabilities in the last decade, with a large percentage of them being identified in just the last few years. (And remember that OS-X is Linux based so Linux vulnerabilities may also impact that OS.) Other vendors with a relatively high number of vulnerabilities include IBM, Cisco, and Adobe. Linux is a slightly different category of software; its code is freely available, and therefore large numbers of researchers (and hackers) can work on both finding and fixing (or exploiting) vulnerabilities. But there are still thousands of identified vulnerabilities in various Linux distributions. A database of identified flaws (a.k.a., CVE—common vulnerabilities and exposure) is maintained by the MITRE organization (https://cve.mitre.org/cve) and you can search that database based on software package, product vendor, or operating system. Similarly, NIST maintains a national vulnerability database (https://

nvd.nist.gov/vuln) that also details the same CVEs and provides a severity rating (a CVSS—Common Vulnerability Scoring System—score), a general description, and (very important) references to possible sources of a patch or update or other mitigation solution. The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of computer system security vulnerabilities. CVSS attempts to assign severity scores, on a scale of 0.0 to 10.0, to vulnerabilities, allowing organizations to prioritize responses and resources according to threat level and risk. A score of 0.0 means no risk at all, whereas a score of 10.0 means a very severe and dangerous risk level. Your organization needs to determine its risk tolerance and decide how to deal with vulnerabilities based on their CVSS value. For example, your patch application policy might require something like this:

SEVERE	CVSS 8.0 to 10.0	Patching within 24 hours
HIGH	CVSS 6.5 to 7.9	Patching within 7 days
MODERATE	CVSS 5.0 to 6.4	Patching within 30 days
LOW	CVSS 0.0 to 4.9	Patching at next outage

Clearly, this is oversimplified, just to make the point that you don't have to treat all patching the same and that you need to give more attention to some vulnerabilities because of the risk they pose, as compared to others. You can also have mitigation alternatives in the event that the patching time window can't be met (e.g., isolate critical systems to which the vulnerability applies if a SEVERE vulnerability can't be patched within 24 hours of its identification). Table 7–2 lists some of the malware that has caused extensive and expensive damage to corporation's computer systems and networks over the past decade or so—good examples of why patching identified security vulnerabilities needs to be a priority.

Table 7–2. Most troublesome vulnerabilities in the past decade

MS17-010 (Eternal Blue)

One of the costliest ransomware attacks in history so far, WannaCry and NotPetya both used Eternal Blue-style attacks as part of their payloads.

CVE-2018-4878

Adobe Flash vulnerability included in multiple exploit kits, most notably the Fallout Exploit Kit that is used to power GandCrab ransomware, which is still active.

MS14-068

This could allow an attacker to exploit a vulnerability in Microsoft Kerberos and elevate unprivileged domain user account privileges.

W32 (MyDoom)

The email worm creates network openings that allowed others to have access to your computer. In addition, the payload also had the ability to open random programs (exceeded the records of the ILoveYou and Sobig worms).

CVE-2019-0708 (BlueKeep)

Despite multiple warnings, it took until November 2019 for the first exploits of Bluekeep to be spotted. It has been predicted that a widespread exploit could be severe.

MS08-067 (Conficker)

This Windows Server Message Block vulnerability is still occasionally seen in older networks with legacy gear, such as exist in the IACS world.

MS01-023 (Nimda)

Nimda was a package of Microsoft IIS exploits that was released a week after the 9/11 attacks.

CVE-2014-0160 (Heartbleed)

Heartbleed is a vulnerability in the OpenSSL code that handles the Heartbeat extension for TLS/DTLS.

CVE-2008-1447 (Kaminsky Bug)

This DNS poisoning vulnerability allowed attackers to send users to malicious Web sites and impersonate any legitimate Web site and steal personal/financial data.

CVE-2014-6271 (Shellshock)

The remote code execution vulnerability in Linux that affected Bash, and could allow an attacker to gain control over a targeted computer, if exploited successfully.

MS02-039 (SQL Slammer)

MS02-039 released in January 2003, causing a denial of service on some internet backbone hosts as well as IT systems and dramatically slowed general internet traffic.

The existence of known weaknesses is why keeping security-related operating system, application, and networking patches up to date is so important. There is always a race going on between hackers, finding yet another hole, and vendors, supplying a plug for (patching) that hole. Some SCADA system owners have taken an attitude toward patches of "if it isn't broken, don't fix it." They are loath to touch anything if the system is working properly. This attitude usually derives from past experience with (or anecdotal stories about) SCADA systems that, upon being patched or upgraded, failed to work properly until the prior version of software was reinstalled. After all, who needs to fear a hacker when the IT group or your vendor is even more likely to shut down the SCADA system? But ignoring the management of security patches is a dangerous risk today. It used to be that a SCADA vendor would review patches and updates from operating system and application vendors and make a determination regarding which patches were or were not essential. Today, that chore falls on the shoulders of the SCADA system owner or their corporate IT organization. Fortunately, there are several credible sources of information about vulnerabilities and patches (separate from the vendors themselves) available as a free reference.

The U.S. government has recognized the need to protect out industrial automation systems and has created groups under the Department of Homeland Security (DHS), including ICS-CERT and US-CERT, and now also CISA, whose task is to help in discovering vulnerabilities, assessing their threat level (and assigning a CVSS value), and also providing patch information, where available (fig. 7–5). An organization can sign up to receive automatic updates from CISA/US-CERT whenever they identify a new vulnerability and then receive a monthly summary of vulnerabilities with high CVSS rankings, including the product and vendor to which they apply. This information allows an organization to determine if any of the vulnerabilities are applicable to their systems, and, if so, how critical it is to get those systems patched.

Another source of information about threats and vulnerabilities, and about how cyberattacks are executed and what tactics are employed, is provided by the MITRE Corporation, a nonprofit research organization located in New England and the D.C. area and founded during the Cold War in the 1950s. MITRE is primarily federally funded and does research for several federal agencies. As such, much of its unclassified work is publicly available, and this includes a recent development: the ATT&CK™ database. This database is an active, constantly updated compilation of actual, documented cyberattack and exploitation techniques, sorted into a dozen categories based on the purpose (discovery, evasion, persistence, etc.) of the technique (fig. 7–6). The categories range from initial asset *discovery* to *detection evasion* during the attack, to establishing a *command and control* channel back from the compromised asset. The database describes specific lists of tactics for each technique and also provides potential cybersecurity mitigations for those tactics/techniques. The database does not provide software, patches, or code; it is strictly informational. The database can be found online at: https://attack.mitre.org.

In addition to the base attack database, which is more IT-oriented, MITRE has also created a specialized version of their database that is focused on industrial automation control system (IACS) technology, and which addresses IACS-specific digital assets and IACS-specific compromise consequences (fig. 7–7). MITRE constantly updates both databases as new attack methodologies and techniques are identified.

Risk Assessment

Risk assessment is the process of determining what threats you feel to be credible and what actions (if any) you need to take to counter those threats. In effect, you are performing risk management wherein you are balancing the cost of countermeasures against the potential impact to employees, the environment, the public, your customers, and your shareholders. There are several methodologies being used in various industries for the purpose of performing a formal risk self-assessment. Some organizations, like the International Society for Automation (ISA), have

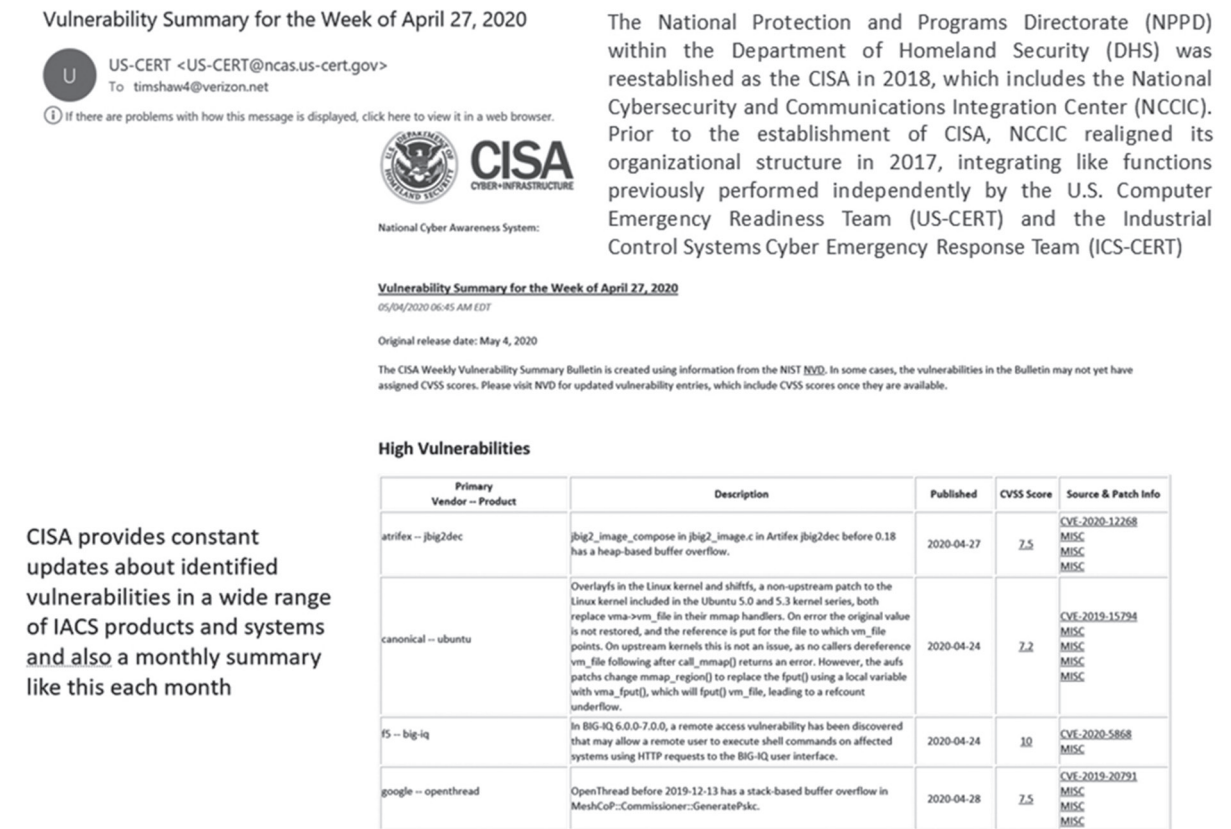


Fig. 7–5. US-CERT/CISA monthly vulnerability updates

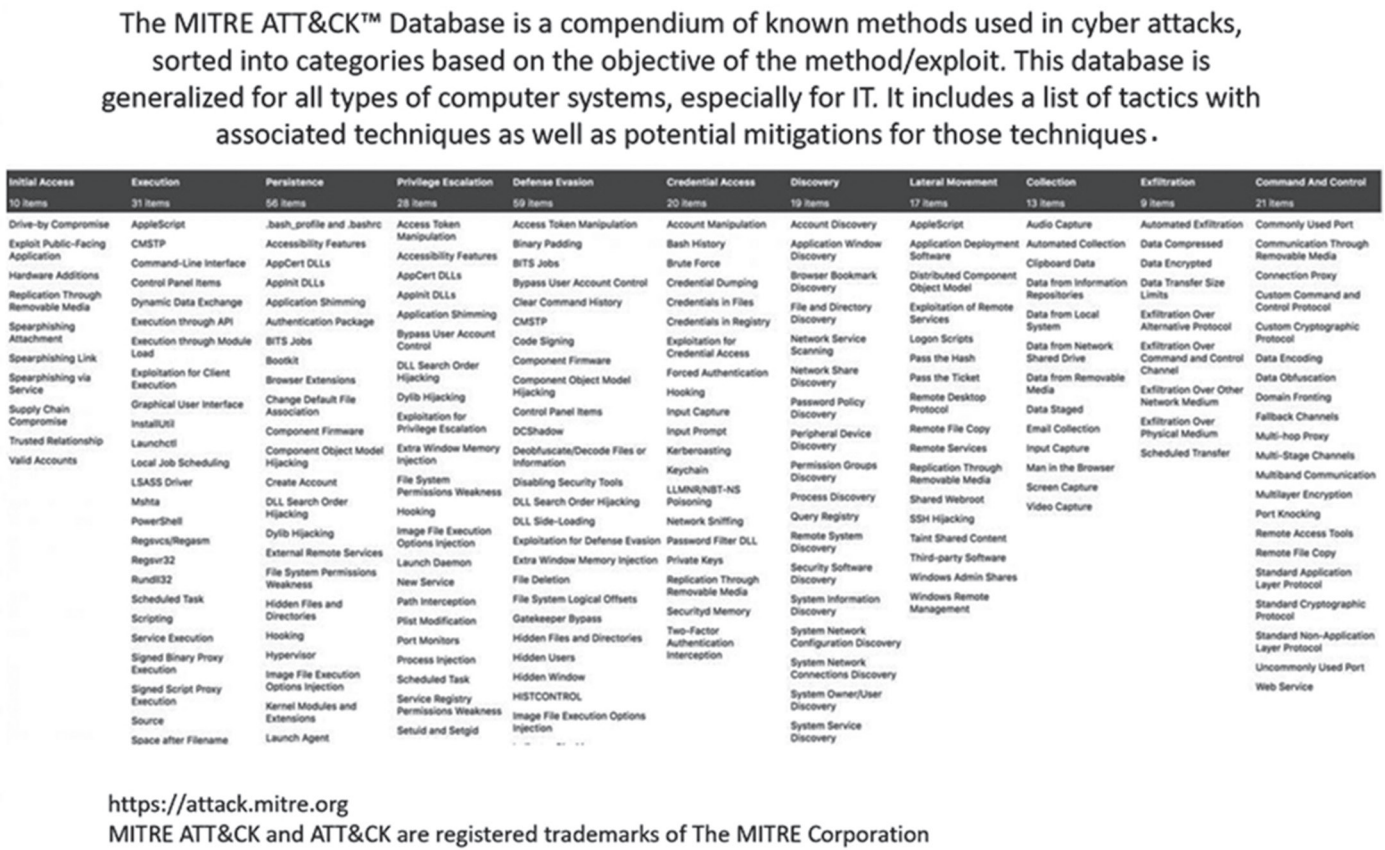


Fig. 7–6. The MITRE Corporation's ATT&CK™ Database

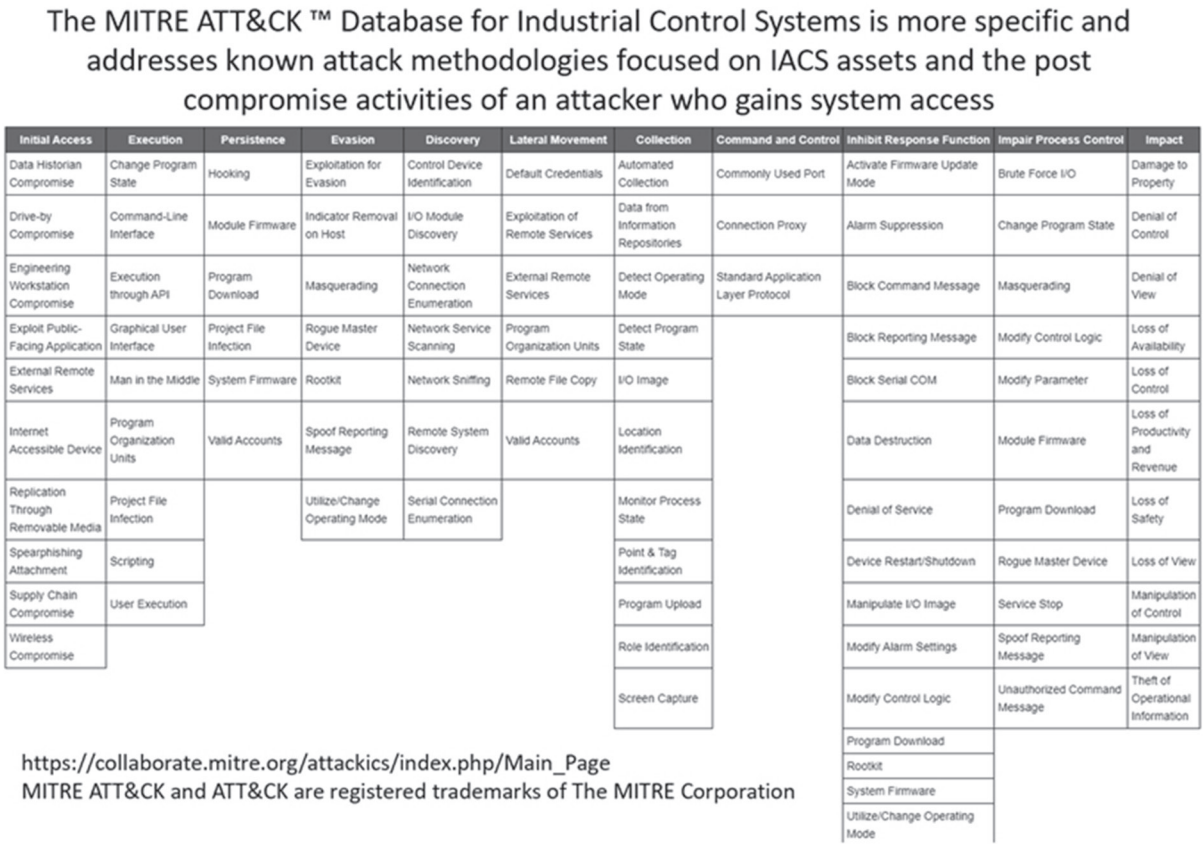


Fig. 7–7. The MITRE Corporation's IACS ATT&CK™ database

published guidelines and suggestions for this process (document TR99.99.01), and governmental organizations, like NIST, have published guidelines as well (NIST SP-800-26 Security Self-Assessment Guide for Information Technology Systems). NIST is an impressive source of information on a whole range of security-related topics. Major corporations, like Dupont, have devised their own methodologies; Dupont formalized theirs with the acronym DNSAM (Dupont Network Security Assessment Methodology). There is no one single way to perform a security self-assessment, and the specific approach you use needs to be tailored to the business methodologies and budgeting procedures and decision-making process used in your particular organization, because, in the end, you are building a business case to justify the manpower and capital outlay (and ongoing annual operating budget) required to implement and maintain the countermeasures you have concluded to be necessary. Of course, if you are in a regulated industry (e.g., domestic Nuclear Power Generation), then you may have mandated physical and cybersecurity requirements and you may feel therefore that there is no reason to perform a risk assessment. The requirements/regulations already dictate what you are required to implement. The problem with that attitude is that 'compliance' with requirements may or may not achieve adequate cyber security. In truth, the earlier requirements imposed by the NERC CIP standards, were pretty minimal and probably not adequate for effective cyber security. Fortunately, those standards have also been evolving and improving.

Remember also that implementing a security program (if you don't already have one in place) is not a one-shot deal; it includes implementing policies and procedures that will have a distinct and ongoing manpower impact. A properly done security assessment needs to address cybersecurity (including communications), but also operational and physical security (which are discussed later in the book). The first step in doing a self-assessment is to put together a team, preferably one with representatives (stakeholders) from various key departments and organizations, including operations, engineering, IT, legal, human resources (HR), etc. Not everyone on the team will be actively involved at every step; some (like legal and HR) will mainly be there to ensure that your plans and suggested policies and procedures don't violate laws or regulations. The next step is to collect information about your critical cyber assets: systems, communications, information, software, facilities, infrastructure, etc. Since this book deals with SCADA systems, it is assumed that we are referring specifically to those assets required for the SCADA system to remain properly operational. Depending on your system architecture, that could include support subsystems such as an historian and/or an Active Directory server—or even other IT assets (particularly if your SCADA system is virtualized). Appendix E provides a set of checklists and tables that can be used for identifying and inventorying these assets. Once the critical assets are enumerated, it is then possible to review them for vulnerabilities, both of a physical nature and of a cybersecurity nature, and to assess what realistic threats exist that need to be countered. The final step (in the self-assessment process) is to select

(and cost justify) the countermeasures being proposed, showing the basis for the decision. Someone once said that perfect security would never be affordable and probably not even possible. What you want to achieve is a level of security that is acceptable, based on the risks and consequence trade-offs.

Vulnerability scanning

Overall, cybersecurity goes beyond individual system elements and needs to consider networks, pathways, protocols, and even procedures—but an attacker is going to specifically attack and compromise individual computers, one at a time, in an effort to gain control over your SCADA system. For that reason, you need to identify weaknesses and vulnerabilities in those computers before an attacker does. The weakness could be from poor configuration or account settings and not just because of unpatched software flaws. The same goes for devices like firewalls and managed switches. If a device has an IP address and supports any form of cross-network communications, then it has the potential to be attacked and compromised. One way to check for weaknesses is to use a commercial vulnerability scanning tool. Automated vulnerability scanning software, such as GFI LANguard™, Nessus™ and Retina E-Eye™, send out huge numbers of specially crafted IP datagrams, plus TCP and UDP packets, to determine which ports and services are present on target computers or devices. They then go further and perform banner grabbing and even execute exploit checks (using a library) to look for unpatched vulnerabilities. These tools can also identify the target's O.S. via "TCP stack fingerprinting," which is watching how it responds to various illegal combinations of TCP message flag bits and checking default TCP settings. The results of these scans are quite comprehensive and include a list of identified vulnerabilities (with the associated CVEs), as well as recommendations for configuration changes that would improve the hardening of the computer/device. Vendors of this type of auditing tool maintain exploit/vulnerability databases and update them as new ones are identified. In essence, these tools are similar to those used by hacker to compromise systems, except they don't deliver a malicious payload when they test an exploit; they merely determine that they could have done so (the vulnerability is there and exploitable). It should be noted that not all systems and devices can handle being given a vulnerability scan, and thus it is never a good idea to perform a scan on a live production system or system component. As an example, various "smart" devices (PLCs, instruments, analyzers) with Ethernet-TCP/IP connectivity have been known to lock up, and require power cycling, when targeted with a vulnerability scan.

In the next chapters, we will discuss hacking and also measures that can be applied to block hackers from getting to your SCADA system.

8

Malware, cyberattacks and hacking tools

As has been mentioned, there are many different kinds of cyberattacks that can be launched against your systems, and there are many types of attackers or threat agents. The most serious threat (a terrorist attack taking control of your SCADA system) is posed by the lowest-probability threat agent. Conversely, the least serious threat (getting infected by an opportunistic virus or worm) is posed by the highest-probability threat agent (random software infection). This is not to imply that a malware infection is not dangerous. Quite the opposite; a really nasty form of malware (e.g., the WannaCry ransomware) can definitely lock up and shut down your SCADA system. But if you take the appropriate precautions with firewalls and NIDS, HIDS and malware-scanning software, your chances of a malware infection are low. If you make and keep adequate software backups and practice recovery and system restoration procedures, the impact and duration of a successful malware infection can be minimized. Here, when we refer to malware, we mean autonomous malicious software that is opportunistic—and not malicious exploits specifically launched against your systems by an active, participating attacker seeking to find and exploit vulnerabilities. As mentioned previously, if you get hit with opportunistic malware, you probably did something wrong and need to review your procedures in regard to portable media, portable devices, Web browsing, and email.

If terrorists chose to attack your systems, they would have most likely gone to great lengths to determine what type of system and computers and operating systems you have and to learn as much as possible prior to launching an attack. If you take steps to prevent them from obtaining critical information (particularly from a former insider), establish and maintain strong and properly designed defenses, and establish procedures and policies that further safeguard your systems, then the chances of a successful attack are small. (Here we are talking about a cyberattack on your SCADA system and not a physical assault on your SCADA facility.) Unlike at the movies, where a cyberattack can cause computer and networks to burst into flames and explode, a real-world cyberattack can, at worst, scramble your systems software and data and cause a crash. If your SCADA system has adequate redundancy (and, most importantly, protections to prevent damage to the primary from spreading to the backup), then you should be able to switch to the backup and then reload and reconstruct the primary system while running in a non-redundant

configuration. If you have an alternative operating facility, then you could activate that facility and transfer operations until such time as you have restored the primary system and site. (Again, it is important that you have protections in place to ensure that damage to the systems at the primary operating facility can't spread to the systems in the alternative facility.) If you make and keep adequate software backups and practice transfer and recovery procedures (to a redundant system and/or an alternative operational site), then the impact of a successful cyberattack can be minimized. If your SCADA system is essentially virtualized, then it is even better, as you can merely delete the damaged VM and start a new clone of the most recent baseline (presuming that the cyberattack did not target the hypervisor or host O.S. within which the SCADA VM is running.) We should again note that no such cyberattack can occur unless the attacker finds a pathway. Since your SCADA system is probably not (and really should not be) connected to the Internet, but likely is connected to your corporate WAN in some fashion, and since phishing attacks on corporations have had an unacceptably high success rate, the most likely and viable attack pathway (ignoring the supply chain and social engineering of portable media for the moment) would be from the SCADA system connection to the corporate WAN. This will be why we will discuss the need for creating a DMZ between your SCADA system and the corporate WAN when we discuss cybersecurity details in Chapter 11. As we just mentioned, one strategy you need to adopt is to ensure that a successful penetration and compromise of one system/computer does not then make it simple and easy for the compromise to spread to other systems and even to the alternate operating facility, if there is one. Just as a DMZ between the SCADA system and the most likely attack pathway (the corporate WAN) is a strongly suggested strategy, placing protective and detective measures in place to block the spread of a successful compromise (e.g., such as a malicious worm like MyDoom) is also strongly suggested. This is why network segmentation using internal firewalls is a strong cybersecurity recommendation. Note that the term "worm" is a description of a particular form of malware that is designed to automatically spread to other systems via network connectivity. The payload that the worm carries defines what the malware will attempt to accomplish, but the worm designation implies that the malware is free-standing and self-spreading, across the available LAN/WAN. We will discuss malware in more detail later in the chapter.

In contrast to a group of motivated and directed terrorists, if a recreational hacker got into your corporate WAN and then stumbled across your systems and decided to probe and play, that person would not know, in advance of breaking in, what types of systems you have or what they are used for. Remember that modern SCADA systems look a great deal like conventional IT systems. To a hacker, such mystery is part of the fun—posing an enjoyable technical challenge. What such an attacker might choose to do, if they were successful in getting into your SCADA system, is impossible to predict. It would be best to keep both attackers out in the first place. One way to be notified of hackers poking around in your systems is to deploy a *honeypot* or *honeynet*.

Honeypot for detection

A honeypot is a simulation of a real system that appears real enough to attract the attention of a person who is probing your systems. Generally, a honeypot consists of servers (probably virtual) and data (for example, in a network site) that appears to be a legitimate part of the site and that seems to contain information or a resource of value or interest to attackers, but which actually is isolated and monitored and enables analyzing the attacker's means and methods of attack. There is free honeypot software available to run as a set of VMs, but you do need to do some configuration work and have a server available (which is preferably not part of your SCADA system) to support the honeypot simulation. You essentially know that there is no legitimate reason for anyone to be probing or scanning the honeypot, and so any attempt to do so can be considered as part of a cyberattack. Honeypots are merely for detection purposes and, if implemented effectively, should look more inviting to attack than your actual SCADA system. Implementing, monitoring, and maintaining a honeypot takes resources, and to use one, you generally need to make it easier to reach than the actual SCADA system, so the attackers hit it first. For example you can logically position it outside your firewall. This can create a potential attack pathway to the SCADA system depending on the approach you take. It must be noted that professional hackers are well aware of honeypot technology and how to identify a honeypot (or at least a virtual machine), so their usefulness is primarily against recreational hackers or hackers of a lower experience level. A honeynet is a similar concept whereby a simulated computer network is created and positioned to be detectable by an attacker. As with honeypots, honeynets are not authorized for any authentic uses. If a honeynet is accessed, a fair assumption is that the person accessing it is unauthorized and potentially malicious.

Hacking tools

We have previously mentioned that there is a wide range of tools that can be used by hackers to attempt to compromise a computer or network. To properly protect against cyber attackers, it's useful to understand the tools and techniques they use. One of the most powerful hacking (or pen-testing, depending on your intentions) toolkits freely available to download from an Internet Web site (https://www.kali.org/) is called *Kali Linux* (previously known as *Backtrack*); this is a Linux distribution containing a huge number of discrete Pen-testing tools, including a package called the *Metasploit framework* (MSF). Kali Linux and Metasploit are both supported by Offensive Security (https://www.offensive-security.com). Natively, MSF requires user interaction with a textual command-line interface (a shell, in Linux parlance), but there is a user-friendly GUI, called *Armitage*, that can be used as a front-end to MSF and that makes using MSF surprisingly easy, even for those with minimal hacking skills (fig. 8–1).

What makes MSF so powerful, useful, and dangerous is that it has an extensive library of exploits (more added each year) crafted for specific operating system platforms and applications, plus a huge library of payloads that can be delivered by

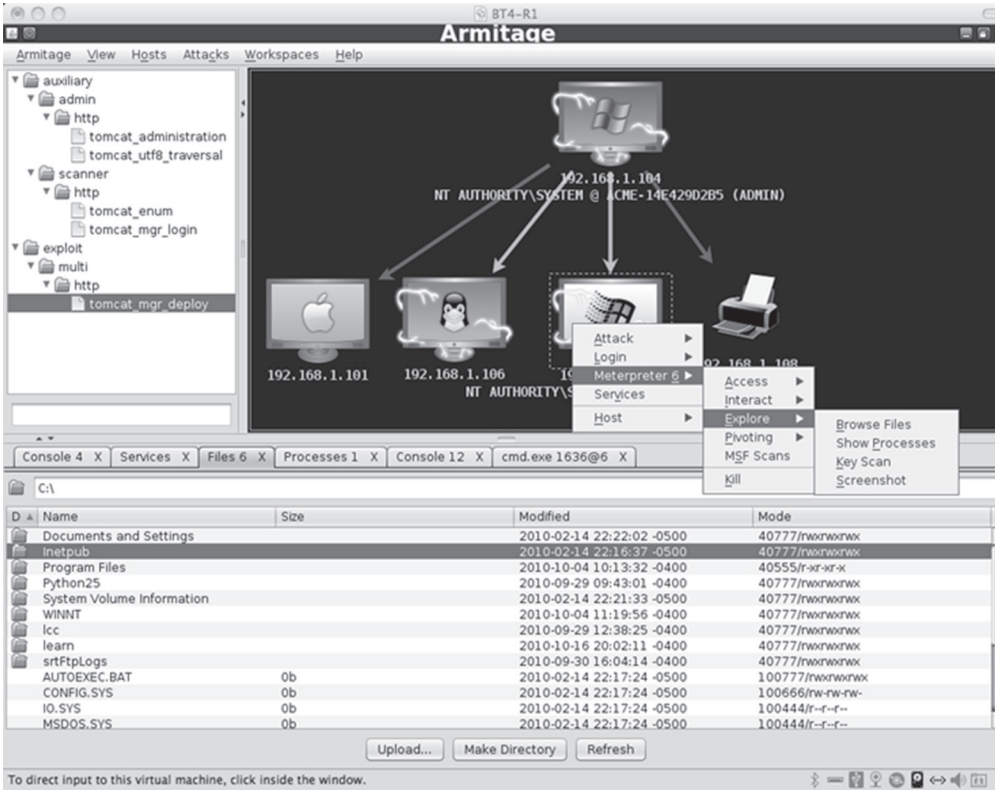


Fig. 8–1. The Metasploit framework with Armitage

these various exploits. In figure 8–2, you can see the first page of a long and extensive list of exploits, and by looking at the first word of the strings in the left column, you can see that the exploits shown there target the *aix, android, apple, freebsd*, and *linux* operating systems, plus many more not shown (such as the long list for various Windows versions—the list is in alphabetical order). When using Armitage, that package will actually make exploit and payload recommendations, from which you may select. The payloads available include non-staged full shell code payloads; stager payloads, where space is limited and where the command and control (C&C) channel will allow the attacker to direct the stager to read in a full payload; the Meterpreter, which is an advanced multi-capability payload that uses DLL (dynamic linking library) injection to hide its presence; the PassiveX payload, which creates hidden instances of Internet Explorer and uses innocuous http messages for C&C (command and control); the NonX payloads, which circumvent Windows data execution prevention (DEP) mechanisms; Virtual Network Console (VNC) payloads, using reflective DLL injection; and even IPv6 payloads. And, of course, if you have the necessary skills, you can develop your own exploits and payloads.

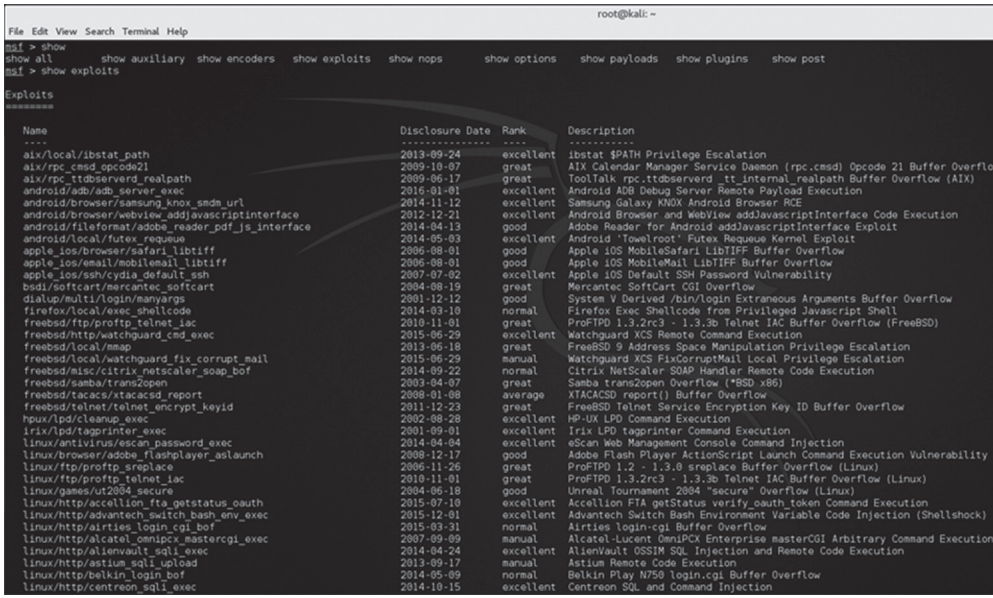


Fig. 8–2. Sample exploits from the Metasploit framework

Using downloaded hacker tools, even unskilled individuals surfing the Internet may mount an attack. There are lots of computer enthusiasts and amateurs who lack the advanced programming skills of a real hacker but, for whatever reasons, seek the thrill of playing hacker games. Those individuals obtain software and do-it-yourself instructions, from various hacker Web sites, on writing viruses and worms and launching a range of attacks. Hackers call these amateurs *script kiddies* because they follow a cookbook script, rather than inventing their own attacks. Hackers speak of these people with disdain yet provide the tools and instructions they use to launch highly destructive worms like ILoveYou and SoBig onto the Internet. Even tools like Metasploit, with its command line interface, can reduce the complexity of staging a cyberattack. A few simple commands and some data entry are all that is required to cause Metasploit to send an exploit and payload against a target machine. In figure 8–3, we can see a typical attack using MSF. The attacker (using a Linux machine) has selected a specific exploit from a list sorted by operating system; in this case, the attacker was able to determine that the target is a Windows XP machine and has selected an exploit that uses a DCOM vulnerability. At the top of the screen, the attacker then sets the desired payload: a reverse VNC server. And then with the command "exploit," Metasploit does the rest and within a few seconds, a new window appears on the attacker's desktop—a window that is displaying the actual desktop of the target machine. At this point, the attacker's keyboard and mouse are now also linked to the target machine and the attacker can type commands and run programs on the target, as if they were sitting in front

of the target computer. Moreover, the attacker is using the account access rights of the person that is/was logged in, who might be someone with Administrative access. And all it took were a half dozen simple commands.

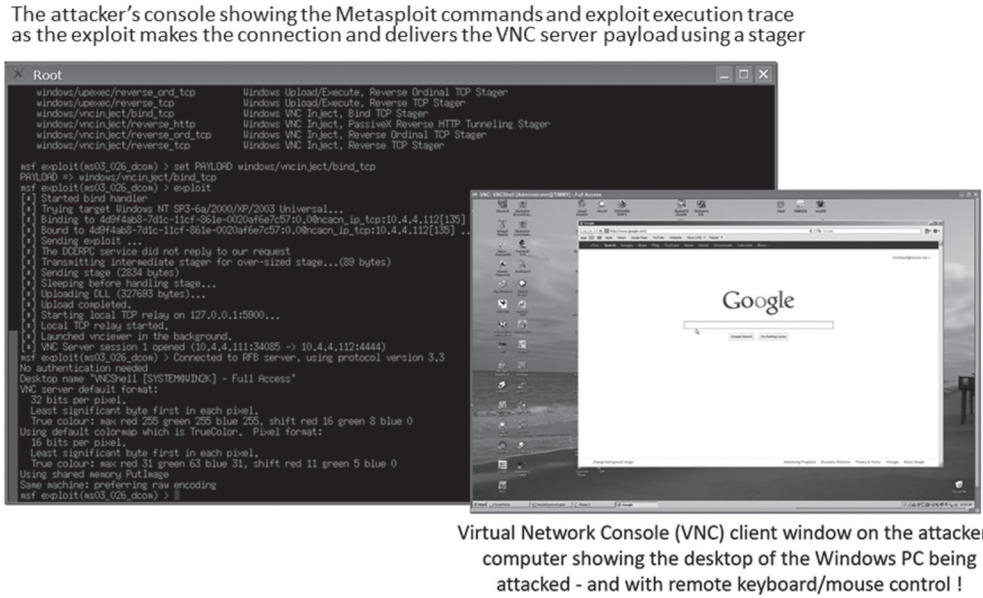


Fig. 8–3. Using Metasploit to inject a VNC into a target computer

As was just mentioned, Kali Linux can be downloaded and used for hacking and penetration testing (fig. 8–4). This distribution comes with a huge number of tools, including MSF and Armitage, which can be used for a range of hacking and pen-testing activities. You never want to perform pen-testing (or vulnerability scan) on a live production system, due to the possibility of damaging that system. But if you have a test environment (or can run a VM of your SCADA software), then making use of the same tools the attacker might use to compromise your systems—to see if you actually can compromise them—allows you to identify vulnerabilities and take actions to eliminate (or mitigate) them. Of course, an attacker with Kali Linux won't be so polite, they will use all the tools available to try to compromise your SCADA system and gain enough access to achieve their objective, whatever that happens to be.

The following table lists the various Kali Linux tools, grouped into functional categories. Most of these tools operate on a shell interface level (command line interface, much like cmd.exe in Windows). This is intentional, as this allows the tools to be run from scripts (BASH or Python, for example) and allows them to be strung together with the output from one being *piped* as input to the next, to create an even more powerful and customized tool. We like to think of the individual tools as being

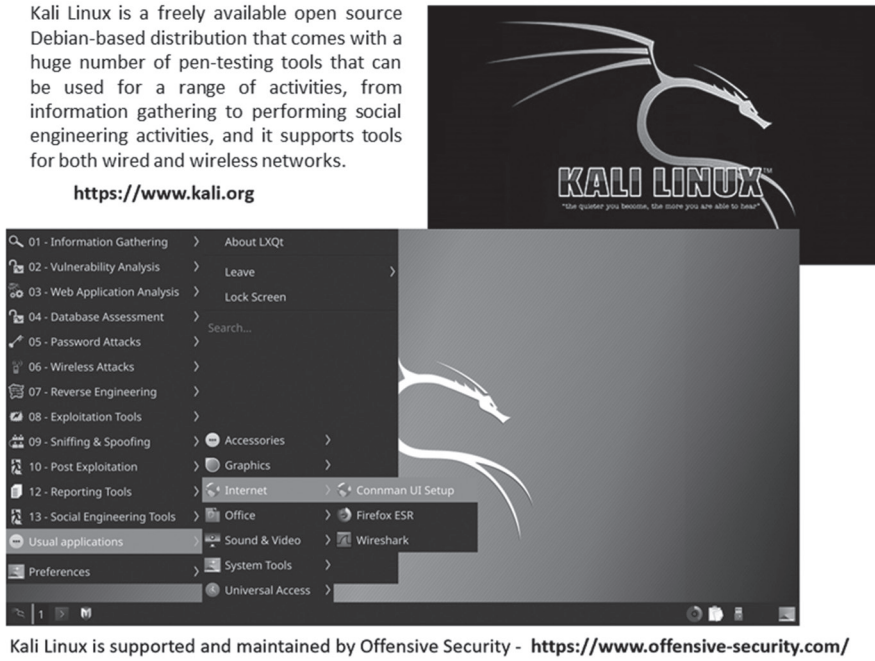


Fig. 8–4. Kali Linux distribution with pen-testing tools

Lego® blocks that can be combined to build a much more powerful hacking tool. The use of scripting also allows an attacker to automate manual activities that would be highly repetitive, such as sending a set of exploits against a large range of IP addresses and TCP/UDP port numbers. The descriptions for all of the individual tools in the following table would fill a separate book, but descriptions for each, including usage examples and command line options (and, in many instances, hyperlinks to additional reference materials), can be found online at: https://tools.kali.org/tools-listing

Table.

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
Information Gathering	ace-voip Amap APT2 arp-scan Automater bing-ip2hosts braa CaseFile CDPSnarf cisco-torch

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
	copy-router-config DMitry dnmap dnsenum dnsmap DNSRecon dnstracer dnswalk DotDotPwn enum4linux enumIAX EyeWitness Faraday Fierce Firewalk fragroute fragrouter Ghost Phisher GoLismero goofile hping3 ident-user-enum InSpy InTrace iSMTP lbd Maltego Teeth masscan Metagoofil Miranda nbtscan-unixwiz Nikto Nmap ntop OSRFramework p0f Parsero Recon-ng SET SMBMap smtp-user-enum snmp-check SPARTA sslcaudit SSLsplit sslstrip SSLyze Sublist3r THC-IPV6 theHarvester TLSSLed

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
	twofi Unicornscan URLCrazy Wireshark WOL-E Xplico
Vulnerability Analysis	BBQSQL BED cisco-auditing-tool cisco-global-exploiter cisco-ocs cisco-torch copy-router-config Doona DotDotPwn HexorBase jSQL Injection Lynis Nmap ohrwurm openvas Oscanner Powerfuzzer sfuzz SidGuesser SIPArmyKnife sqlmap Sqlninja sqlsus THC-IPV6 tnscmd10g unix-privesc-check Yersinia
Wireless Attacks	Airbase-ng Aircrack-ng Airdecap-ng and Airdecloak-ng Aireplay-ng airgraph-ng Airmon-ng Airodump-ng airodump-ng-oui-update Airolib-ng Airserv-ng Airtun-ng Asleap Besside-ng Bluelog BlueMaho Bluepot BlueRanger

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
	Bluesnarfer Bully coWPAtty crackle eapmd5pass Easside-ng Fern Wifi Cracker FreeRADIUS-WPE Ghost Phisher GISKismet Gqrx gr-scan hostapd-wpe ivstools kalibrate-rtl KillerBee Kismet makeivs-ng mdk3 mfcuk mfoc mfterm Multimon-NG Packetforge-ng PixieWPS Pyrit Reaver redfang RTLSDR Scanner Spooftooph Tkiptun-ng Wesside-ng Wifi Honey wifiphisher Wifitap Wifite wpaclean
Web Applications	apache-users Arachni BBQSQL BlindElephant Burp Suite CutyCapt DAVTest deblaze DIRB DirBuster fimap FunkLoad Gobuster

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
	Grabber hURL jboss-autopwn joomscan jSQL Injection Maltego Teeth Nikto PadBuster Paros Parsero plecost Powerfuzzer ProxyStrike Recon-ng Skipfish sqlmap Sqlninja sqlsus ua-tester Uniscan w3af WebScarab Webshag WebSlayer WebSploit Wfuzz WhatWeb WPScan XSSer zaproxy
Exploitation Tools	Armitage Backdoor Factory BeEF cisco-auditing-tool cisco-global-exploiter cisco-ocs cisco-torch Commix crackle exploitdb jboss-autopwn Linux Exploit Suggester Maltego Teeth Metasploit Framework MSFPC RouterSploit SET ShellNoob sqlmap THC-IPV6 Yersinia

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
Forensics Tools	Binwalk bulk-extractor Capstone chntpw Cuckoo dc3dd ddrescue DFF diStorm3 Dumpzilla extundelete Foremost Galleta Guymager iPhone Backup Analyzer p0f pdf-parser pdfid pdgmail peepdf RegRipper Volatility Xplico
Stress Testing	DHCPig FunkLoad iaxflood Inundator inviteflood ipv6-toolkit mdk3 Reaver rtpflood SlowHTTPTest t50 Termineter THC-IPV6 THC-SSL-DOS
Sniffing & Spoofing	ettercap Burp Suite DNSChef fiked hamster-sidejack HexInject iaxflood inviteflood iSMTP isr-evilgrade mitmproxy ohrwurm protos-sip rebind

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
	responder rtpbreak rtpinsertsound rtpmixsound sctpscan SIPArmyKnife SIPp SIPVicious SniffJoke SSLsplit sslstrip THC-IPV6 VoIPHopper WebScarab Wifi Honey Wireshark xspy Yersinia zaproxy
Password Attacks	BruteSpray Burp Suite CeWL chntpw cisco-auditing-tool CmosPwd creddump crowbar crunch findmyhash gpp-decrypt hash-identifier Hashcat HexorBase THC-Hydra John the Ripper Johnny keimpx Maltego Teeth Maskprocessor multiforcer Ncrack oclgausscrack ophcrack PACK patator phrasendrescher polenum RainbowCrack rcracki-mt RSMangler SecLists

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
	SQLdict Statsprocessor THC-pptp-bruter TrueCrack WebScarab wordlists zaproxy
Maintaining Access	CryptCat Cymothoa dbd dns2tcp HTTPTunnel Intersect Nishang polenum PowerSploit pwnat RidEnum sbd shellter U3-Pwn Webshells Weevely Winexe
Reverse Engineering	apktool dex2jar diStorm3 edb-debugger jad javasnoop JD-GUI OllyDbg smali Valgrind YARA
Reporting Tools	CaseFile cherrytree CutyCapt dos2unix Dradis MagicTree Metagoofil Nipper-ng pipal RDPY

Tools Included in Kali Linux Distribution	
Functional Category	**List of Tools**
Hardware Hacking	android-sdk apktool Arduino dex2jar Sakis3G smali

NOTE:
Many of the tools listed can be used for more than one purpose and thus will appear in the table in more than one category. In earlier versions of Windows, in order to utilize Kali Linux, you would need to download the Kali distribution and use it to create a Linux virtual machine (VM) that could then be run using free virtualization tools such as VMware player© or VirtualBox©. Interestingly, as of Windows 10, Microsoft actually supports an application (*Kali for Windows*) that allows you to run the Kali Linux tools from within Windows, without the need to install and run a virtual environment such as VMware or VirtualBox. The Windows Subsystem for Linux (WSL 2) package allows users to run Linux applications directly on Windows. The application can be downloaded for free from the Microsoft Web site at: https://www.microsoft.com/en-us/p/kali-linux/

The WSL package (currently at version 2) can be enabled on Windows 10 several different ways, but the simplest way is to use a PowerShell command (while logged in as an Administrator), then restart your computer, and then install Kali Linux. Microsoft kindly provides directions on their Web site: https://docs.microsoft.com/en-us/windows/wsl/install-win10

It is useful to discuss the various categories of tools (what they are used for) with the Kali groupings, as many of the tools are associated with one or more of the MITRE ATT&CK© techniques:

- Information gathering—These tools would be used for performing the discovery aspects of the MITRE attack chain, at least for discovery as regards the actual computers and network structure. But discovery also includes social engineering. A toolkit for performing social engineering, called SET (social engineering toolkit—of course), is also part of the Kali Linux tools. But the list of tools in this section ranges from network mapping tools to firewall rule discovery tools, and Wireshark is included in this set as well. APT2 (automated penetration testing) is another tool, and it will perform an NMAP scan and generate a list of recommended exploits to use against the identified hosts. There are also tools specific to certain hardware platforms, such as Cisco devices and tools for exploring Windows workgroups.
- Vulnerability analysis—This category is somewhat what MITRE would call weaponize; it consists of tools for identifying the exploitable weaknesses in the identified hosts and networks. Of note, this group includes tools for *fuzzing* (applications that let attackers or pen testers send large quantities of invalid, unexpected, or random data as inputs to other programs in order to discover bugs or hidden functionality).

The tools also include some for identifying SQL injection vulnerabilities. Yersinia can be used to attack weaknesses in layer 2 protocols, such as DHCP.

- Wireless attacks—This set of tools helps you break the encryption on WAP and WEP access points and gain wireless access. Also, tools to help you identify clandestine APs (those not broadcasting their IDs) and tools to set up a spoofed AP.
- Web applications—These tools let you attack Web sites and identify weaknesses in Web servers and Web applications. Tools such as Foca and Skipfish will actually upload and dissect an entire Web site automatically, looking for useful information. Tools such as WebSlayer let you craft spoofed http GET and POST messages to test Web site user input error checking.
- Exploitation tools—This group corresponds to the MITRE exploit phase and includes, of course, the Metasploit Framework and Armitage. But it also includes tools for attacking routers and switches, and this is where we again find SET, which can be used to create spear phishing attacks and a malicious Web site. There are also tools designed to scan a Linux system to identify vulnerabilities.
- Forensics tools—There is no correspondence with the MITRE attack chain and these tools, as these are for after-the-fact forensic analysis. These are not attack tools but tools you would utilize if your system were successfully compromised, in order to discover how it happened, what was the vulnerability they found, and how did they exploit that flaw. Dumpzilla can, for example, extract all forensically interesting information from the Firefox, Iceweasel, and Seamonkey browsers so it can be analyzed. The Binwalk tool is designed for identifying files and code embedded inside of firmware images. There are even tools for pulling forensic information from phones.
- Stress testing—In general, these tools generate large amounts of message traffic in order to overload networks and services. They don't specifically correlate to a MITRE attack phase, but if you were attempting to blind a NIDS to your actual attack, or attempting to crash a service, these tools would come into play. The FunkLoad tool will generate massive Web site traffic. The Inundator tool is specifically designed to anonymously inundate intrusion detection (IDS) logs with false positives to obfuscate a real attack.
- Sniffing & spoofing—These tools would correspond to both the deliver and discovery phases of MITRE, as they are used to gather information as well as generate message traffic intended to bypass protections such as firewalls. Ettercap is used to perform ARP cache poisoning as well as staging man-in-the-middle attacks (which can be used for credential

capture and other purposes). SIPVicious can be used to scan and audit VoIP phones. The responder tool is designed to spoof responses for Windows file server (SMB) requests and redirect the requester to the attacker's computer.

- Password attacks—This set of tools specifically deals with cracking passwords and includes tools for doing that, such as John the Ripper, which can break Windows password hashes, and Hashcat, a password recovery tool for Linux, OS X, and Windows. Hashcat can break Microsoft LM hashes, MD4, MD5, SHA-family, Unix Crypt formats, MySQL, and Cisco PIX hashes. It should be noted that for brute force password cracking, you also need word files (which are available on the Internet for free), and this group includes one: wordlist. This tool set also includes RainbowCrack, which uses rainbow files to almost instantly break password encryption.
- Maintaining access—This relates to the MITRE command & control phase, as these tools help you hide your presence, establish mechanisms to keep a backdoor in place, and create a covert communication channel between you and the compromised system.
- Reverse engineering—These tools are similar to the forensic tools mentioned above. Tools like OllyDbg are used to debug machine code to uncover what it is doing—useful for determining what malware is designed to do when there is no source code. YARA is a tool for identifying and classifying malware.
- Reporting tools—Includes a range of tools for examining and reporting, such as Nipper-ng, which will read a set of firewall rules and generate a report on issues it identifies.
- Hardware hacking—These tools are really more for experimentation with developing programs (possibly malicious Apps) for specialized hardware platforms, such as an Android phone or an Arduino single-board microcontroller.

As you can see, although Kali Linux does provide a number of tools useful for information gathering and machine compromise, it also contains a lot of tools designed to be used by the good guys for testing their own systems for vulnerabilities and for researchers.

The threat actors have some seriously powerful tools to use in their efforts to attack your SCADA system. But with properly established and maintained protective measures and the use of technologies such as firewalls, NIDS, and HIDS, the chances of a professional or recreational hacker (and certainly a script kiddy) gaining access into and playing with your systems while remaining undetected is quite low.

Vulnerabilities

When we speak of identifying and exploiting vulnerabilities, what does that actually mean? First it is important to understand that there are two broad categories of exploits that take advantage of identified vulnerabilities: *local exploits* and *remote exploits*. A local exploit is typically a specially designed program that will need to be loaded onto the target system, compiled (unless it is a script), and executed. You need access to the facilities of the computer to do this and at least a user account with basic access rights. A very common form of local exploit is one that is used to gain administrative access in a process called *privilege escalation*. If an adversary can compromise a normal, low-level user account, then they may attempt to use a local exploit to get an administrative (root in Linux) account. At that point, the adversary has unlimited access and control. In basic Windows workgroups, an Admin-level user on one machine is essentially granted the same rights on the other machines over the network—professional courtesy of sorts. Local exploits typically target system services and driver components since these often run at an administrative access level. Local exploits, like all exploits, are designed to attack identified weaknesses in the program they are attacking.

The other broad category is the *remote exploit*. These are attacks that take place over an IP network (LAN or WAN) connection and involve services that are intended to accept connections from clients and have some form of interaction (communication exchange) with those clients. In this case, the attacker may, or may not, require an account on the system they are attacking, or just knowledge of a vulnerability in the service. Earlier in the chapter, we discussed drive-by download attacks by malicious hacker Web sites. That is a form of remote exploit in that the Web browser (the Web client) accepts and blindly executes the mobile code sent to it by the attacker. Actually, the first thing a browser does after setting up a TCP session and exchanging header information is to ask the Web site for its default home page. Some unpatched versions of IE were, at that point, vulnerable to a buffer overflow attack. Sadly, no user interaction was required by the person whose computer was attempting to connect to the Web site. Basically, moments after clicking on a hyperlink in a phishing email, your computer was compromised—generally, with no indication of this happening.

The main reason for vulnerabilities in most software is due to either poor programming practices (possibly by others when publicly available libraries are used in your own programs) or due to inherent weaknesses in the programming language used. The C (and C++) programming languages were the most commonly used languages in the 1980s and even 1990s, and unfortunately those languages have built-in weaknesses in some of the standard library functions. Also, in C (C++), there are data structures (e.g., arrays) with no inherent bounds checking. It is up to the programmer to write bounds-checking logic, which many never did. It is quite possible to have a program that expects to receive

up to "x" bytes of data, but which will keep receiving as long as the data keeps coming. With TCP/IP, a program expecting a message from another program essentially says, "here is where to place the data; please wake me when it has been received," and then goes to sleep. Clearly, the receiving program has no control over and is not monitoring what is actually received, and TCP will keep delivering as long as the sender keeps sending data.

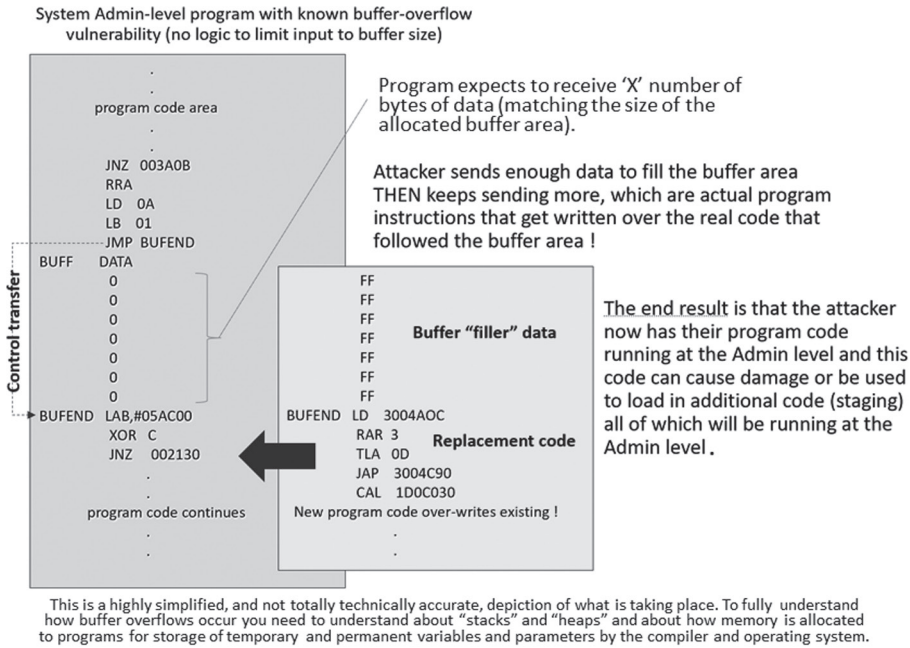


Fig. 8–5. Buffer overflow attack

In a buffer overflow attack, the space allocated for the incoming data (the buffer area) is filled up with some random bytes, but then actual machine op-codes (instructions) follow and are written into memory, replacing the legitimate instructions in that memory area—essentially loading the attacker's program into executable memory. If properly done, when the program "wakes" after the data is received, the code it resumes executing is that of the attacker. And that code is executing at the priority level of the program that was just altered, which, if a system service, is probably at the root or Admin level. Often the amount of code space available to be overwritten is quite limited, and so hackers work to develop the smallest payloads they can (least number of machine instructions) that still accomplish the essential tasks. Often a small payload (called a *stager*) merely makes a call to the operating system to make an ftp (file transfer protocol) file transfer request from the attacker's computer and then makes a call to the operating system to run the program that was just uploaded—which need not be tiny, as

the operating system will allocate disk space and memory to that program, just like any other program. In fact, it might be a remotely controllable copy of Metasploit (the Meterpreter payload).

A technique used in local exploits, which is similar in concept to a buffer overflow attack, is a technique called s*tack smashing*. Stack smashing attacks often make use of a hardware design weakness in x86 CPUs where a dynamically allocated memory buffer (called a stack) is used to hold both program data and memory pointers. When a program calls a library function, the CPU jumps to the memory where that function code is located. When the function completes, it needs to jump back to the calling program. To do this, the calling program's return location is pushed onto the stack. Programs (and functions) can also store temporary data on the stack, where the assumption is that programmers will be careful about putting data onto the stack and removing all of their data—but only their data—from the stack when the program terminates. Stack smashing is breaking the rules and programming code that writes instructions over the stack area owned by other programs (processes) as well as the "return" addresses of those processes (fig. 8–6). Typically, a function called by a process will cause the *return address* (where to jump back into the process when the function is done) to be placed on the stack. But in stack smashing, the attacker overwrites that address value, and when the function returns, it now jumps to the malicious code placed on the stack, which, as with a buffer overflow attack, is probably a stager program. Stack smashing requires loading and running a program on the machine and having an account that allows you to do this, so it is generally only used as a local exploit, often for privilege escalation purposes. There are techniques, such as using *stack canaries* and designating the stack memory as non-executable (it can hold data but not program instructions). Both approaches are system-level strategies (and may depend on advanced CPU capabilities) and not something done specifically by programmers in the way they write their programs.

Programming language compilers implement the stack canary feature by adding code at the entry point of function definitions—code that stores a special value on the stack (the stack canary) during the function prologue—and then adding code to pop off the stack and check the canary value in the epilogue of the function, prior to attempting a return to the calling program. If the canary value was changed, then the added compiler code calls a failure handler rather than attempting to return to the calling program.

Current INTEL and AMD microprocessors have a special hardware feature, a flag in the CPU (XD-Execute Disable or NX-No Execute), that can be set by the operating system to designate that some memory pages are for data only, and the CPU will refuse to execute instructions in those code pages. Both the Windows and Linux operating systems can take advantage of that hardware feature and use data-only memory for the stack, so that stack smashing will fail.

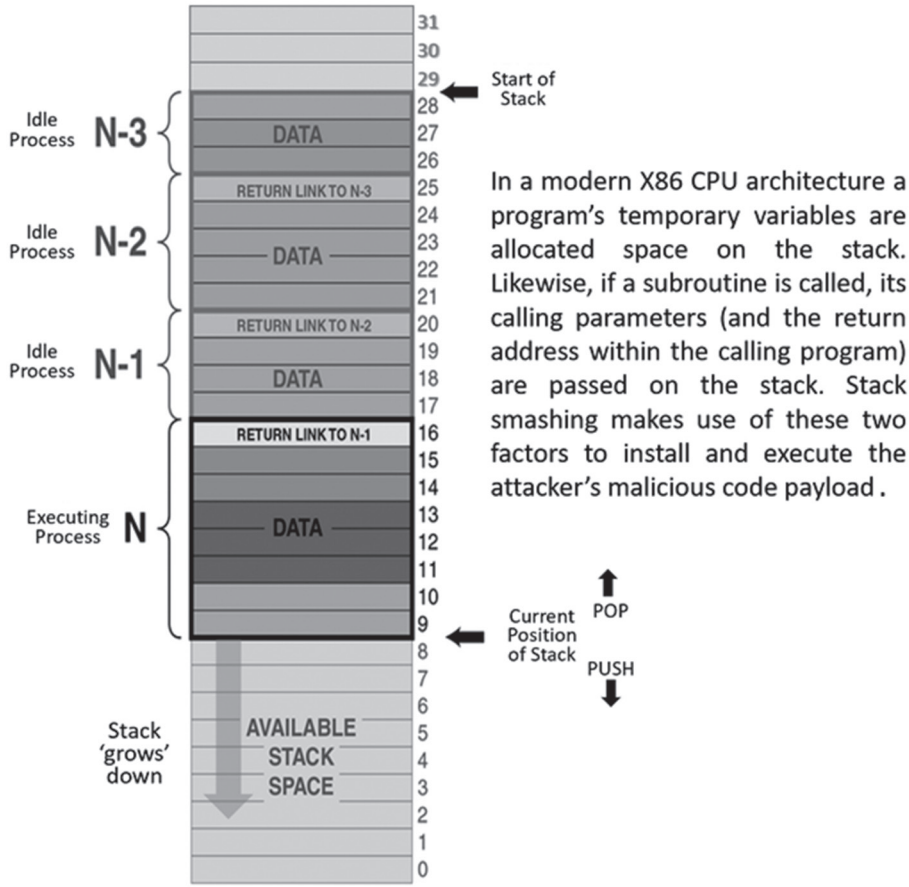


Fig. 8–6. Stack smashing in an x86 CPU

WEB Server/SQL Injection

Although most SCADA systems don't (and really shouldn't) have actual Web servers offering up real-time data directly onto the Internet, a fair number use Web technology for the operational HMIs on their LAN and for internal data presentation and access. A good number also interact, through intermediate relational databases, with corporate Web servers that offer information within the corporate WAN and possibly the Internet. Those Web servers may possibly also support XML data exchanges with the equivalent business-to-business (B2B) Web servers of major customers, suppliers, and partners. Some SCADA systems use relational databases for their historical data and even for system configuration data. Web servers that interact with back-end relational databases to obtain or store requested

information are vulnerable to an attack technique called *SQL injection.* This type of vulnerability isn't of the same category as we have just discussed, but it is an exploitable weakness created by poor programming practices. In this technique, specially formulated malicious text is entered into data entry fields on Web pages (Web forms), such that the logic behind the Web page (usually some server-side script written in a language like PHP, VBScript, or Perl) is tricked into executing an SQL command that is embedded in the entered text (fig. 8–7). In effect, the specially constructed text, entered into the data entry field, has the effect of modifying the query made to the relational database constructed on the server side, using the user-entered text. This only succeeds because of inadequate user-input validity checking on the client-side Web page (e.g., using JavaScript). The result of SQL injection could be to cause confidential database tables and their contents to be revealed, or to cause the alteration or destruction of the tables and their data.

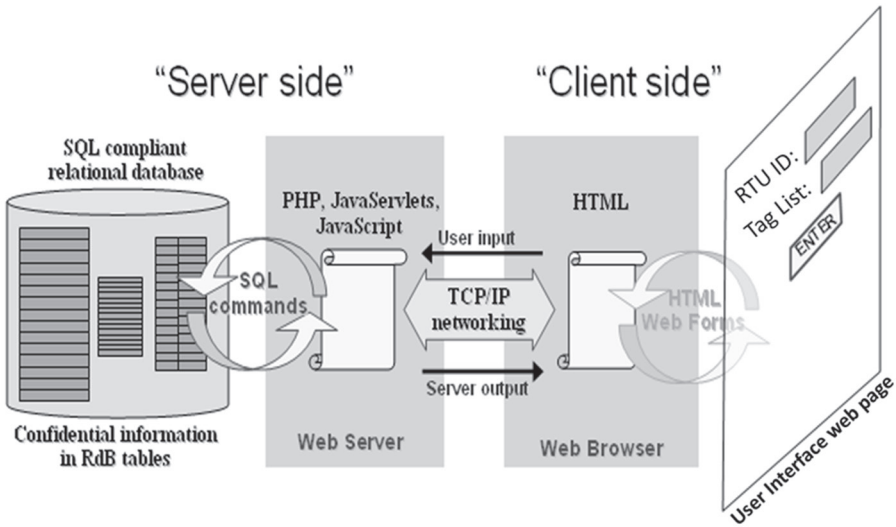


Fig. 8–7. Relational database tampering via SQL injection

SQL injection is successful mainly with Web pages that have poor (or no) validity checking on their data entry fields. This is basically a vulnerability created by poor programming practices on the part of the Web site developer. If a person's name is expected, then an input like "John Bleecker; UNION SELECT Card_No FROM CARDLIST WHERE ' ' = ' ' " is probably an attempt at SQL injection and should be rejected (and preferably a log entry should be sent to the SIEM package or syslog server). SQL injection vulnerability can be a concern if you make use of Web interfaces internally to access back-end databases. If an attacker manages to get into the corporate WAN, they may discover internal Web servers and attempt SQL injection attacks, possibly to seek confidential information, or just to generate damage.

Fortunately, there are commercial products that can scan your Web pages and identify places where an SQL injection attack could be used (including some of the tools in Kali Linux). Addition of suitable input validity checking to the Web pages will generally prevent SQL injection attacks.

Email and Web Browsing

Phishing attacks have already been mentioned as being one of the primary ways in which cyber attackers gain a foothold in computers within your corporation (which then serves as a platform for digging deeper into your networks, possibly reaching the SCADA system). Phishing, and all of its variations, depends on your being willing to click on a hyperlink provided within the body of the email, an attachment or a web page. If well done, meaning the message content is very convincing and believable, phishing can (unfortunately) be surprisingly successful. But another means of attack is to provide a malicious email attachment (some form of document) that contains malware, usually in the form of a *script*, that will be executed when you open the attachment (this approach is also called *letter-bombing*). This is different from an attachment that just contains a hyperlink—which is just a variation on phishing. With a letter bomb, the document is usually opened by an application, such as Adobe Reader, Office Word or Excel, or a Web browser. Those different applications support various scripting languages, which support differing levels of capability. All of the Microsoft Office applications (including Internet Explorer, but not Edge) support Microsoft's VB (Visual Basic) scripting language (at least up till late 2019, when an update disabled it in IE). VB scripts have extensive power to manipulate system resources and objects, such as the file system and networking. (IE also supports JavaScript, as do other Web browsers, but JavaScript has more restrictions on what it can do.) Someone once described VB script as "the hacker's best friend" because of the almost unlimited power it provides. You can embed VB scripts in all of the Microsoft Office application documents (in that case, it is referred to as VBA—Visual Basic for Applications), such as Word documents, PowerPoint presentations, and Excel spreadsheets. It can also be embedded in Web pages (as mobile code or server-side scripts) because IE has a VB interpreter, as does IIS. (But none of the third-party Web browsers support VB scripting, just Microsoft.) VB script makes use of the Microsoft Component Object Model (COM) to access the various elements of the system. Although, going forward, Microsoft plans to eliminate VB scripting, it has a huge installed base and will be given legacy support for some time to come. A VB script can, depending on where it is running, create and delete user accounts, manipulate files, manipulate Windows services, run applications, and even manipulate system settings (including Windows firewall and network settings). For those reasons, it is good to take preventative actions such as disabling the execution of VBA in all of the Office applications and also in your IE Web browser. Your corporate

IT department should have already taken those steps for all company computers, but, if not, then you should do it yourself. Preventing an attacker from getting into the corporate WAN is keeping them one step further away from the SCADA system. Because of the possibility of infected documents being spread inadvertently (including via portable media), you also want to take these same steps on any computer that is part of the SCADA system or a connected support system if they use any Microsoft Office applications and/or Adobe products.

Adobe documents for both their Reader and Acrobat application can have JavaScript code embedded in the document. Adobe Reader and Acrobat come with support for JavaScript embedded in the PDF file. Hackers can simply add malicious JavaScript code to the PDF file to exploit this vulnerability. When you open the PDF file with Adobe Reader or Acrobat, the malicious JavaScript gets executed and installs malware on your computer, all without your knowing it. Of course, there are limits on what JavaScript can do—which is far less than what is possible with VB scripts and ActiveX objects—but it is enough. A recent, and unfortunately effective, form of ransomware was totally written in JavaScript. Again, there are security settings in the Adobe products that can disable embedded JavaScript execution.

Web browsing without adequate protection is another way to provide a hacker with a means for compromising your computer and getting onto the corporate WAN. There is a huge and constantly changing array of malicious Web sites on the Internet (and here we are not talking about the *dark web*, just the normal World Wide Web). These Web sites are often sites offering pornography, free online gaming, or gambling. But they can also be sites that seem harmless, such as sites catering to selected hobbies or interests. What is even worse is that an estimated 82 percent (per the Naked Security Blog by Sophos—https://news.sophos.com/) of the malicious sites on the WWW are actually legitimate Web sites that have been hacked and made malicious, unknown to the Web site owners. (In many instances, the only thing done to the legitimate Web site is to cause it to redirect your browser to the attacker's Web site.) When such a site is visited, the site will automatically attempt to push *mobile code* (such as VB script or JavaScript) or malicious *ActiveX objects* into your Web browser and use that foothold to further infect your computer. (This is called a *drive-by attack* or a *drive-by download* attack because, due to how browsing works, no user action is required to initiate the attack, other than browsing the Web site.) ActiveX, another Microsoft technology, is the name given to small control programs (mobile code) that are sent to your Web browser as part of rendering a Web page, so in many aspects, they are like VB scripts. The ActiveX control files are downloaded and stored to a default folder on the computer hard drive. ActiveX is still supported as of Windows 10 through Internet Explorer 11, while ActiveX is not supported in their newer default Web browser: Microsoft Edge (which has a different, incompatible extension system). ActiveX has been used to create Web page objects with intelligence and animation (like that month

calendar that appears and lets you pick a day), but as they can use COM to access system elements, they can also be dangerous and malicious. Microsoft has taken steps over time to improve the cybersecurity of ActiveX (originally, all you got was a pop-up note asking if you wanted to let the control be loaded, with minimal information about its source or purpose—what Microsoft called *Authenticode*), but it has proven to be a technology often abused by hackers. Again, it only works with Microsoft IE, and there are security settings within IE to block loading and execution of ActiveX controls and objects. A common cybersecurity suggestion has been to use another browser because of ActiveX support in IE. Microsoft (in both IE and Edge), and all other Web browser vendors, support JavaScript in their Web browser applications. While VB script and ActiveX are Microsoft proprietary technologies and only work in IE, JavaScript is an open standard that works with all browsers. By the way in which Web browsers/servers and the http (hypertext transfer) protocol work, it is rather easy to attempt to send malicious scripts to your browser:

1) Your browser sends a http GET request to the Web site/server, asking for the site's default Web page.
2) The hackers will have crafted an "index.html" Web page containing references to mobile code (or ActiveX objects) they have developed, and they send it back to your browser in response to the GET.
3) Your browser starts to parse and render the Web page and
4) sends one or more requests back to the malicious Web site, requesting it to send over the mobile code/ActiveX objects.
5) And once they are received, your browser begins executing their malicious code and you are compromised. It was all invisible to you and nearly instantaneous.

In some instances, the hackers will not use Web pages to deliver malicious content. As your browser has had to set up a TCP session with their Web server, they can use that persistent connection to attempt to probe your computer for exploitable vulnerabilities and then hit it with exploit code as the means for delivering malware as a payload.

Malware

Malicious software (a.k.a., malware), in this instance, is intended to mean software that either spreads itself or attaches to other software in order to live and spread, but in neither case is there a human attacker involved in its spread or function, once developed and introduced into the first system. Malware can be categorized in several ways: by how it propagates or reproduces, by how it is executed, and by what it does. We have just been discussing two forms of malware based on scripting languages, both of which totally depend on being loaded and executed by some application program with a scripting execution engine. Without that environment, they are

nonfunctional and harmless. *Viruses* are a form of malware that must infect another program in order to be executed and then attempt to spread itself to other systems. A virus is generally designed to infect and embed itself into a specific program and so one way to reduce vulnerability to many viruses is to harden you computers by eliminating unnecessary programs that may be the known target of a virus. Clearly, a virus will have an underlying function to perform as well, even if just to reproduce and consume all resources (that form of virus is also called *bacteria*). A *worm* is a form of malware, similar to a virus, because it spreads itself, but unlike a virus, it doesn't infect other programs. Instead, it is a standalone piece of malware that spreads from one system to another or from one network to another via available LAN/WAN IP-network connectivity. Again, a worm will normally have an underlying function to perform. So, the terms virus and worm really just describe two categories of malware that live and spread differently. A Trojan horse, sometimes just called a Trojan, is any malware that pretends to be something else but really serves a malicious purpose. For example, a Trojan might appear to be a free game or a useful utility, but once it is installed, it might destroy your hard drive, steal data, install a backdoor, or take other harmful actions. Trojans don't typically propagate themselves; they depend on their seemingly useful functions in order to get people to pass them around. Often, we just describe malware based on what function it performs, from harmless but annoying (e.g., Adware) to dangerous and clandestine (e.g., a rootkit). The following is a list of malware descriptive names based on what the malware does, regardless of whether it lives and spreads as a virus, worm, script, or Trojan:

Descriptive Name	Malicious Function(s)
Adware	Adware is a type of malware that downloads or displays advertisements to the device user. Usually, it is more of an irritant in that it forces users to see ads that they would rather not have on their system.
Backdoor	A backdoor is a secret way to get into your device or network. Vendors may create backdoors into their products either intentionally or unintentionally through sloppy coding practices. Backdoors can also be installed by other types of malware, such as viruses or rootkits.
Bots and botnets	A bot is remote-control software installed on your computer to allow an attacker to use your computer as part of a Botnet. Attackers often use botnets to send out spam or phishing campaigns or to carry out distributed denial of service (DDoS) attacks against Web sites.
Browser hijacker	A browser hijacker changes the behavior of your Web browser, for example, by sending you to a new search page, changing your home page, installing unwanted toolbars, directing you to sites you did not intend to visit, and displaying unwanted ads.
Crimeware	A general term for malware that is used to commit a crime, usually a crime that results in financial gain for the attacker.
Keylogger	A keylogger records all user keystrokes, including emails and documents typed and user IDs and passwords entered for authentication. Attackers use this type of malware to obtain passwords so that they can break into networks or user accounts.

Descriptive Name	Malicious Function(s)
Malicious mobile app	General term for apps from the Google/Apple stores that turn out to have malicious functionality (like a Trojan).
RAM scraper	RAM scraper malware harvests data that is being temporarily stored in a system's memory, or RAM.
Ransomware	The most common malware variants lock up a system by drive encryption until the victim pays a ransom to the attacker, usually in the form of Bitcoin.
Rogue security software	A.k.a. scareware. It tricks users into thinking that their system has a security problem and entices them to pay for a fake security tool to fix the problem, but the fake security software often installs more malware onto their system.
Rootkit	Rootkits give attackers administrator-level access to systems without the users' knowledge. Once an attacker has root access to a system, they can do almost anything they want with the system, and such malware can even hide itself from detection.
Spam	Spam is unwanted email (and, per se, not malware), but often comes with hyperlinks or attachments that would install malware on your system.
Spyware	Gathers information about someone without their knowledge or consent. For example, Web site tracking, ID and password recording, collecting email addresses, or looking for credit card numbers or other financial information.
Timebomb	Malware, possibly a Trojan, that waits for a trigger event (often, time/date) before initiating its malicious activity.

The term *payload* is used to describe the function(s) performed by malware, which generally fall(s) into one of the categories listed above. Some forms of malware incorporate both a worm and a virus propagation mechanism and may have multiple malicious functions. Some malware attempts to create and maintain a *covert communication channel* back to the hacker's computer (a.k.a., a command and control or C&C channel), so that the hacker can upload more tools and directions and download captured information. As part of this, they may install a *port listener* and register it to a high-order ephemeral port number. (Netcat and Cryptcat are two very popular port listeners—and they can *port forward* too, which may be needed for creating a C&C connection.) In a situation where a firewall blocks access to a service on a computer that you want to reach, if you can install Netcat on that computer and connect it to a port not used and not blocked by the firewall, then you can have it receive messages and repeat them to the service you needed to connect with. That is called port forwarding, and it is useful for getting around firewalls. Today, some forms of malware are polymorphic, meaning that they actually make changes to themselves as they propagate and reproduce, in order to avoid detection by signature-matching AV tools.

Clearly, some of these categories of malware would be unlikely to cause serious harm to a SCADA system or actually be able to survive and perform their function if they did manage to get into the SCADA system. Spyware designed to find and

exfiltrate credit card information would waste away in a SCADA system. If you have reasonable protective/detective mechanisms on any pathway out to the corporate WAN, and thence to the actual Internet, any attempt to create and maintain a backdoor or C&C pathway or to upload stolen information should be blocked, or at least detected. Even the now infamous Stuxnet malware would do little harm to a SCADA system, unless you happen to have IP networking to the field and use Siemen's PLCs as your field devices.

We did mention at the beginning of the chapter that we were, for the moment, ignoring malware delivery via the supply chain and by means of portable media and digital devices. Malware that adopts a worm format propagates via IP network connections, and so can a virus, but, more often, a virus spreads outside of the system where it is residing, by the movement and copying of infected files—which could occur across a network connection, with some help, or, possibly more likely, on portable media and infected laptop PCs. So to fully protect your SCADA system from malware, you will need to address not just any network connectivity between the SCADA system and other systems/networks, you will need to consider how to protect against its propagation on portable media and mobile digital devices.

In Chapter 11, we will return to a discussion of cybersecurity and how to employ and apply technical countermeasures. But as has been mentioned, cybersecurity assumes the existence of at least adequate physical security, so this is the topic of our next chapter.

9

Physical security

The topic of internal threats and insiders as potential attackers has already been mentioned, and as much as we would like to believe that it couldn't happen, there are far too many examples that prove it can. As already stated, an insider is a person who has some level of authorized physical access to your facilities and who may also have some level of authorized electronic access to systems (possibly the SCADA system) and access to information that could aid in staging a cyber (or physical) attack. If we assume that an insider is a threat agent, then one of the ways to defend against such a threat is to use physical security mechanisms and measures. As the events of 9/11 demonstrated, terrorist groups are willing to make physical attacks on chosen targets, even if this means losing their lives. Suitable physical security measures would be needed to defeat or at least blunt such an attack.

A basic tenet of security is the concept of least privilege, whereby a person is assigned only the access rights required to perform his or her duties. Access rights are supposed to be reviewed routinely on a periodic basis and also whenever a person's duties are modified. For example, just because Fred was allowed SCADA control room access last week, to fix a wiring problem, doesn't mean that such access permission should continue after he has fixed the problem. Unfortunately, for a variety of reasons, personnel tend to collect access rights over time and never lose them. If someone decided that Fred was to be trusted in the control room once, obviously he can be trusted there again, so why go through the effort to change his access rights? In many organizations, physical security is rather loose. Once you make it past the guard at the front door, nothing prevents you from going wherever you please. Social engineering experience has shown that *tailgating* into an access-controlled facility and then walking around boldly and being obvious makes you seem more legitimate, rather than sneaking about quietly (and it helps to pick up a coffee cup and some paperwork and pretend to be talking to someone on your cell phone—all social engineering ploys).

Physical security deals with the physical protection of critical systems and their interrelated subsystems, as well as the protection of confidential and critical materials (including information in a physical form), of personnel, and of actual facilities. Today, physical security even extends to the protection of portable or mobile devices owned by the corporation. This book is focused on the security of critical cyber assets, specifically those that are part of a SCADA system. Thus, we

will not address issues of physical security relating to natural and manufactured disasters or issues not immediately relevant to securing the critical cyber assets—even though these threats still need to be addressed. (A SCADA system can be knocked out by a hurricane or a facility fire just as completely as by a cyberattack.) Physical security programs are intended to prevent the following sorts of things from occurring:

- Harm to personnel
- Theft of assets
- Physical damage of assets or facilities
- Compromised system integrity and function
- Unauthorized disclosure of sensitive, critical, proprietary information
- Service/production interruptions
- Loss of, or damage to, production/product
- Environmental release/damage

In setting up a physical security program, you need to consider priorities, as well as limited funding and resources, and apply them in an optimal manner:

- Security issues you must address:
  - Required by law or regulation
  - Safety (i.e., fire) codes
  - Required to mitigate legal liability/risk
- Security issues you ought to address:
  - Low-cost moves that provide material benefits (locks, chains, etc.)
  - Making/maintaining "backups" of critical systems
  - Having and practicing incident response procedures and recovery plans

Your physical security program might also need to deal with issues such as fitness for duty: ensuring that personnel who have critical jobs are drug- and alcohol-free when they arrive for work. It may also have to deal with ensuring that personnel do not bring dangerous items such as weapons or chemical agents (or infected portable media) into the facility.

Access Controls

The starting point for most physical security is to control access to your facility, which includes your buildings and outbuildings and possibly even the parking lot. The events in Oklahoma City in 1995 demonstrated that a vehicle filled with explosives, allowed to park too close, can take out a building and everyone in it. Most organizations now use *bollards* and other physical barriers to keep vehicles at a distance. There are also a wide range of gates (automated and/or manually operated) that can be used to restrict and control both vehicle and personnel access. The term gate includes things such as turnstiles within a building, possibly

activated by a keycard or PIN (*personal identification number*). Properly designed gates can prevent both vehicle and personnel piggybacking (fig. 9–1).

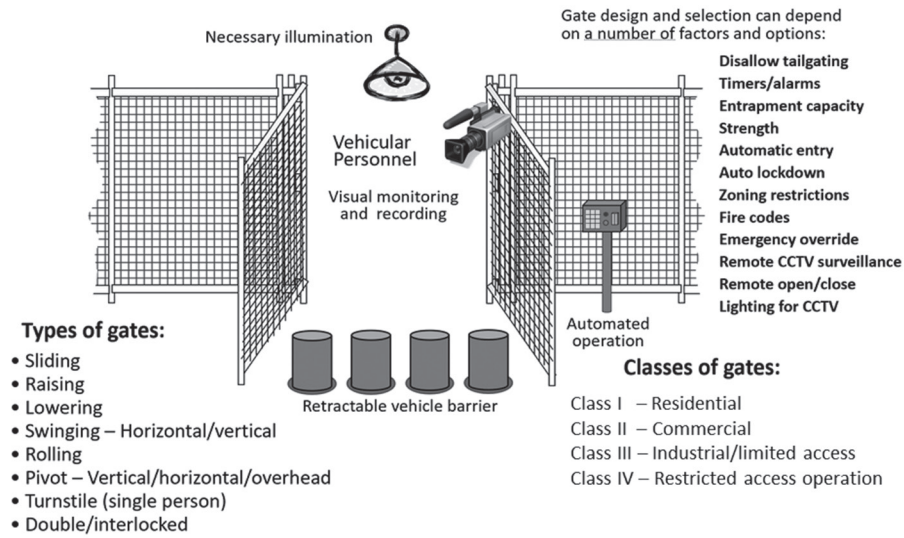


Fig. 9–1. Gates and fencing to control vehicle and personnel access

On the assumption that different personnel will have different levels of access to facility areas, there needs to be a means for enforcing those restrictions. The first level of access control is at the entrance(s) to the facility. SCADA systems are often housed in facilities with multiple purposes. It may be a corporate headquarters, an engineering facility, or just an office building shared by multiple organizations. That being the case, there needs to be public access, and you cannot prevent people from coming and going at will. It is therefore generally necessary to create one or more layers of physical protection around your critical cyber assets, internal to the facility itself, to address this issue (fig. 9–2). Key card systems, physical locks and keys, and even security personnel (a.k.a. guards) can be used to control and restrict access to areas needing protection. Clearly, the SCADA system control room should be access-restricted, but so should the server room and the telecom room.

NERC describes the need to establish a physical security perimeter around critical systems, for automation systems used to run the power grid. This is not always easily done, especially with consideration for fire codes and personnel safety, but it is important to place a physical barrier between unauthorized personnel and your SCADA system and critical subsystems. Hallways can be blocked with one-way, solid-core doors. Windows can be secured with wire mesh, bars, or even shatter-resistant polycarbonate laminates.

The important issue is to carefully review the physical security perimeter and watch out for such weaknesses as suspended ceilings, large air ducts, raised "computer

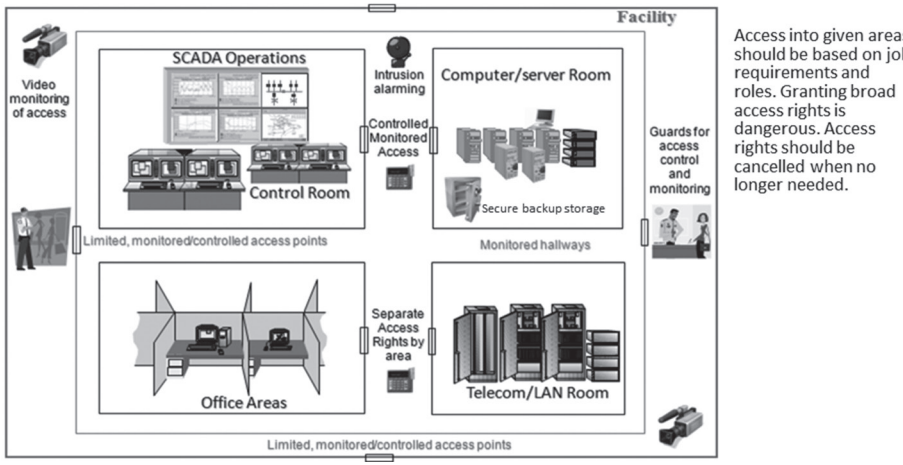


Fig. 9–2. Physical security layers for added security

floors," and other inconspicuous ways in which your perimeter can be breached. The objective is to limit the number of entry points (without compromising personnel safety) so that you can then establish access controls (and monitoring) at those few points of potential ingress. However, because of fire codes and personnel safety, you may not be able to restrict egress via the same small number of locations; you cannot require people to go through authentication procedures if the fire alarm is going off. It is usually assumed that people leaving a restricted area are not as big a concern as people entering, since they must have had access rights to enter in the first place. Of course, if an insider were trying to remove critical equipment or materials or make a quick getaway, the best approach would be to pull the fire alarm and head for an egress point (ideally, one that is not monitored). Access-control mechanisms themselves can be totally manual, totally electronic, or a hybrid of both.

Manual access controls

At the lowest level, in many organizations, manual access control amounts to using locks and keys (possibly electronic) as a means for restricting access to only authorized personnel. If a person has a key, they are obviously authorized. This is a dangerous assumption and lacks a critical component: an audit trail of who entered and exited and when. And let's face facts, keys can be stolen, most can be duplicated, and locks can be picked (although there are solutions for these issues). Today, there are innovative key/lock schemes to address this problem. You can use mechanical cipher locks, which have a keypad, but don't use a key; you enter the numeric combination and the lock opens. There are also keys that are supposedly non-replicable, locks designed to be un-pickable and also high-tech locks that use biometrics (fingerprint scanning). If you want to have an audit trail of access to various rooms, and reduce the number of keys being issued, there are also

key-management systems where personnel check keys in and out and use a unique PIN to get into the key storage unit, which makes a time-stamped recording of each access. Figure 9–3 shows examples of some of these technologies.

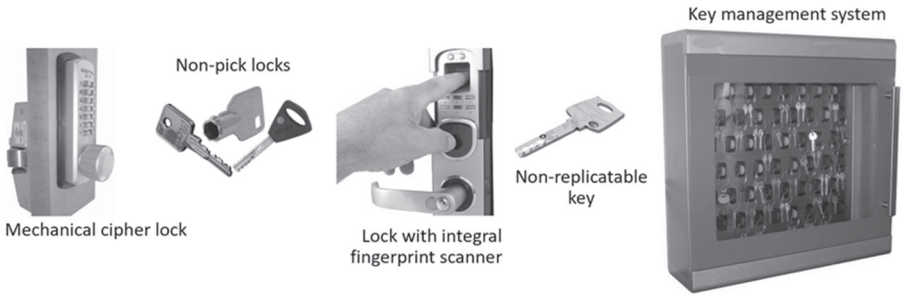


Fig. 9–3. Various forms of high-security key/lock systems

An acceptable totally manual system would employ guards to check credentials and refer to access-control lists (and who have been trained on the company's access policies and procedures), to determine if a given person is allowed entrance beyond that point (and under what conditions they may bring materials in and out and escort unauthorized people). Non-forgeable credentials are a very important provision in such a scheme, so that the guard can be assured that the person offering the credentials must have gotten them legitimately. Most states now issue driver's licenses that are extremely difficult to forge or modify. They contain holograms, photographs, and florescent dyes that stymie the counterfeiter. Employee credentials need to be of the same caliber, particularly if verification is strictly through visual inspection by a guard. This is because the credentials are all the guards can use as the basis for their decision (to allow entrance) and because guards are often low-paid or temporary employees and thus frequently changed. Under such a system, the loss of credentials and especially the theft of credentials need to be treated as serious matters, because an attacker may be trying to obtain credentials to modify or to learn their features and construction. Don't forget that the vendors who sell you the equipment to produce your credentials (ID cards, badges, enrollment systems, etc.) will sell the same equipment to others.

Electronic access controls

Since employment of guards has both pluses and minuses (mainly, overall cost and retraining issues when guards are replaced), many organizations have switched to fully electronic access controls. We have already covered many of the technologies used for personnel authentication by a computer system: passwords, biometrics, magnetic-strip cards, smart cards, RFID, and so forth. These are generally a superset of the same technologies used for electronic remote access authentication.

A computer today can look you in the eye and then look at your stored photographic image and decide whether they match. And this works unless you change your hair style, are wearing a hat, or get contacts to replace your glasses. In the meantime, while we wait for facial recognition technology to become more reliable, you need electronically readable credentials of some sort. Since we want little or no chance of forged or stolen credentials being used for entrance, multiple authentication factors will normally be required: a magnetic or smart (RFID) card plus a password or code number (a PIN). Biometric authentication would be the best in theory, but as mentioned, there are still reliability issues with these technologies. (Biometric scanners usually employ some form of recognition threshold that can be set higher or lower. If you get too many false positives, you can raise the setting; if you get too many false negatives, you can lower the setting.) Granting people access to sensitive areas incorrectly is unacceptable but refusing access to authorized personnel is also a problem. Biometrics tend to work best in a hybrid arrangement, in which a human guard can use other authentication methods to resolve issues of false negatives. Noncontact RFID smart cards with an integrated photographic ID (and possibly a manually entered code/PIN number) are the most prevalent technology used today for fully automated, electronic access control. However, as biometric technologies improve and decrease in cost, expect these to gain market acceptance and wider general usage. One reason for using a PIN, in addition to other factors, is the ability to have a duress code that can be used to indicate a need for a security response (e.g., the "visitor" being escorted is actually holding a knife on the employee).

Automated access controls

Many secure facilities make use of fully automated electronic access-control systems for both general access and for specific access into restricted areas and rooms. In an automated access-control scheme, rooms are secured with safety doors that are locked with either a mechanical, solenoid-operated bolt, or via powerful electromagnets operated by a local access-control panel, which also interfaces to the card/badge reader, possibly a keypad, possibly an audible alarm of some type (e.g., bell, klaxon, siren), and proximity switches that sense door closure (or non-closure). The local control panel is much like a special-purpose RTU, just with different forms of inputs and outputs. (In fact computer-based access monitoring and control systems are very similar in design to SCADA systems. Some vendors of these technologies actually started as SCADA vendors.) The local control panel may, or may not, contain a database of access-control data that tells it who is permitted access, when, and for how long. Some systems use centralized authentication, but that can be a problem if you have communication issues or the central system is unavailable. Pushing access control information out to the local control panels eliminates these issues. Figure 9–4 shows a simplified example of how an automated security door might be equipped.

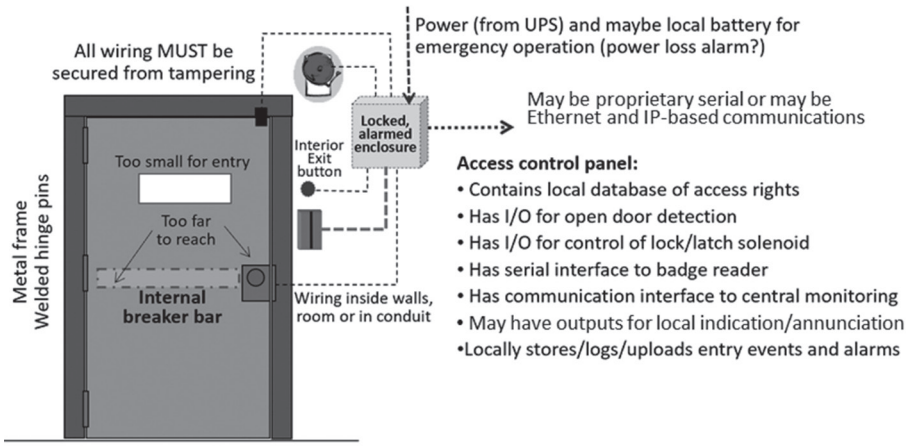


Fig. 9–4. Typical electronic access-control door

An important factor in an automated access-control system is ensuring that it can't easily be circumvented or disabled. If wiring is readily accessible or if the door can be taken off its hinges or if the door has a large breakable window, then access control is too easily compromised. In a small facility, each area requiring access control could be equipped with individual, stand-alone control panels with audible alarms. But it is more typical to connect all of the alarm panels into an integrated system (like a SCADA system for physical security), and this enables both centralized alarm monitoring and centralized access-control database management and personnel (dis)enrollment (fig. 9–5). If personnel are terminated or quit, change jobs, or are hired, then the access-control system needs to be updated and credentials either deleted or enrolled in the system. Many newer access-control systems today use Ethernet-TCP/IP networking to connect the central monitoring station and all the control panels that are downloaded with access-control lists, which provide localized alarm detection, user authentication, and access-control information. Older systems may use proprietary vendor protocols and vendor-specified communication wiring, but the latest such systems can coexist on a shared, switched Ethernet LAN, possibly using a unique VLAN (and, of course, a physically private LAN could be used if preferred). These systems don't merely control access, they also detect and alarm on unauthorized access (i.e., physical intrusion detection), such as a forced door, and can even warn if a door remains open too long after an authorized entry. Some systems also allow smoke/fire sensors and motion sensors to be incorporated as inputs, as well as integrating video/audio surveillance equipment. Your cyber security program may need to (probably should) include adding detective and protective measures to the physical security system as well as the actual SCADA system since compromise of the physical security system could enable a physical attack on the SCADA system.

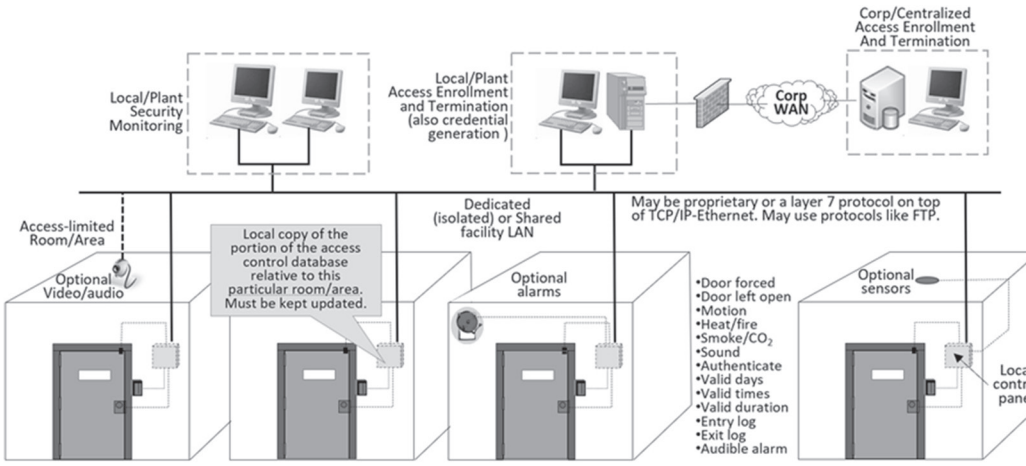


Fig. 9–5. Electronic, automated access-control and intrusion-detection system

Video surveillance

An alternative to having an automated access-control system is to use security officers (guards) to monitor access points and perform credential checking. But rather than having a large number of guards located around the facility, it is more typical to deploy video surveillance equipment at key locations and have a centralized guard station where officers monitor all of the video feeds and from where an officer can be dispatched if something unusual or improper is noted. Such a centralized system may also support remote control of door/gate locks so that a guard makes the permit/deny decision. Video systems today can incorporate a wide range of advanced capabilities such as motion detection and even facial recognition. And such systems can record the video stream for later playback and as a visual record for legal or investigative purposes.

Tamper detection

A malicious insider can also take advantage of their physical access to systems and equipment to cause damage, steal information, or to plant malware in computers. A threat actor who gets into the supply chain may also have that same opportunity or may just attempt to deliver malware by intercepting a physical delivery of software (on a DVD/CD) or equipment (new laptop PCs) and infecting them. In most cases, you will not be able to continuously monitor everything you would like, and you will have to accept that there will be "holes" in your security that might allow a malicious insider to tamper with your important stuff. Your next best option is to at least have some means of identifying potential tampering and/or unauthorized access. This can often be accomplished through the use of inexpensive

tamper-indication devices, such as tape, seals, and clips that can't be removed without obvious signs (damage) of this having been done (fig. 9–6). Tamper seals can be applied to cabinet and enclosure doors and enclosures themselves to show that someone has entered or opened the cabinet since it was last sealed and checked (most tamper seals/tape have unique serial numbers that allow you to identify that they have been replaced with a different seal/tape). Port locks (a.k.a. *port blockers*) can also be used to identify unauthorized connectivity to USB and Ethernet ports. Anti-tamper devices may not prevent tampering (although often they do), but they let you know it has occurred so that you can take appropriate action.

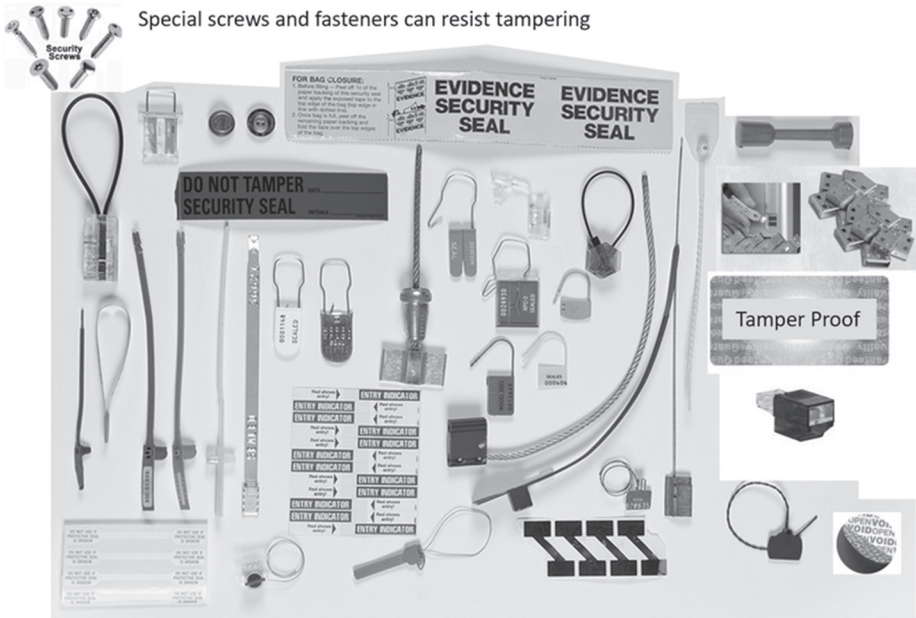


Fig. 9–6. Tamper-indicating/detection mechanisms

Access Tracking

Security professionals will also generally recommend that you keep track of the entrance and exit of personnel who have access to secure areas. Just as a NIDS or HIDS package identifies cyber threats by watching for unusual and anomalous activities inside your computers and on your networks, keeping an audit of personnel access can aid you in spotting anomalous personnel behavior, which might be an indication that an insider is planning something.

Access tracking can be manual or electronic. If a guard is the means for access control, a sign-in log is often the means of access auditing. With almost

any electronic access-control technology, the systems used maintain an electronic log file for auditing purposes. Both types of logs, physical and electronic, should be considered as containing sensitive and confidential information and accorded appropriate protections. A physical logbook would provide a potential attacker with names, departments, and typical access times/days for authorized personnel. An electronic log could contain a lot more data, including employee ID numbers, telephone numbers, and other credential-related personnel information. A potential attacker would be aided by getting their hands on such information. A hostile insider would be aided in covering up illicit activities by destroying such information. Procedures should be developed to secure the access logs and protect the information they contain—for example, making multiple daily backup copies (including photocopies) and storing them somewhere secure (even offsite).

Illegal-Entry Alarms

Access-control mechanisms keep out the honest people, but the dishonest will try to circumvent such controls—by getting the necessary credentials through illicit means, disabling (or attempting to bypass) the access-control mechanism, or by finding holes in your perimeter (an unsecured window, a service tunnel, a false ceiling, etc.). We have already discussed using unforgeable credentials to prevent their illicit use if lost or stolen. But what about people gaining access through other unauthorized means?

From a security standpoint, it is good practice to place sensors (separate from the access-control mechanisms) on doors, windows, and other access ways that can detect their opening—particularly if such access ways cannot be visually monitored. An illegal-entry alarm is triggered when such an access way is used but no access-control authorization was given (and who authorizes access through a window anyway?). Another scheme is to use motion sensors in rooms and areas containing critical equipment or materials and to generate an alarm if motion is detected when no authorized access has been granted to the room or area.

The idea is to make it as difficult as possible (preferably impossible) for any potential threat agent to gain access to your critical systems and confidential materials and information. If they do gain entry to those areas, you want to detect this immediately, so that you can minimize the time available to them to inflict damage or access materials. A very important procedure for an organization with critical systems and information is to develop, test, and practice responding to illegal-entry alarms—and ensuring that everyone knows their role in such a situation.

Physical Isolation of Assets: Layers of Defense

In the design of the physical security perimeter that will protect key computing systems, personnel, and confidential materials, it is wise to follow the old dictate not to place all your eggs in one basket. It is less than ideal to create one, big enclosed area and assume that once inside, you and all of your systems and confidential materials are secure. It is also probable that there will be differing levels of access authority required for subareas within your perimeter. It is still fairly standard to stage large computer racks filled with servers and their various peripherals in a computer room with environmental controls, raised flooring, and a UPS. The separate environmental controls require that this area be enclosed and isolated from the general office environment and have limited points of ingress/egress. With this design as a basis, it is easy enough to add electronic access controls on the entry points and to require a higher level of authorization to enter the computer room than is needed to enter the general office area. This same approach can often be taken with the operational consoles and associated display and annunciation equipment in the control room. These are usually placed in a physically separate control room with its own access ways.

Physical Protection of Materials and Information

Owning and operating a SCADA system to monitor and control your distributed processes isn't the same as being a government agency in charge of spying on enemy agents. You're not likely to have sensitive information of a top-secret level lying around. However, you do have sensitive information like your personnel IDs and passwords, drawings and designs related to your communications system, backup copies of the system configuration files, and many other such materials that need to be kept away from the unauthorized and protected from theft, destruction, or tampering. Potential attackers would be greatly aided in their efforts by such information. Surprisingly, you may also have information that is considered relevant to the Sarbanes-Oxley (SOX) act of 2002 (which addresses all information related to business processes) and which, therefore, needs suitable protection mechanisms, audit trails, and access controls, lest you and your organization be subject to the fines and legal prosecution allowed under that law. (It would be quite unusual for a SCADA system to contain and present information relative to the health insurance portability protection act [HIPPA] but you ought to make sure!)

In accord with the layered defense strategy outlined above, physical materials—whether documentation, manuals, drawings, or computer-readable media—need to be protected as well and housed in storage that affords some level of access

control. Locked cabinets can be effective in keeping people without adequate authorization (as indicated by being issued a key or combination) away from materials to which they have no access rights.

There ought to be procedures for tracking materials and ensuring their timely (and unaltered) return when they are removed from safe storage. If materials are very confidential, it may also be necessary to take approaches to ensure that they cannot be duplicated without proper authorization or discarded without ensuring their adequate destruction (shredding paper documents, smashing disk drives or CD/DVDs to bits, etc.). For materials essential to recovery from a system failure (e.g., backup tapes or disks), merely locking up a set of media in a cabinet in the computer room is probably not sufficient (although it is a good start). The best practice for handing such essential materials is to keep multiple backups, made before and after any software or configuration changes to the system, possibly going back a couple of updates.

Since restoration of the systems from these backups will be necessary if the systems suffer a hardware failure or a software corruption (possibly due to a successful cyberattack or because of a failed update), it is important to know that the backup images will actually be usable and correct (which means that they need to be tested and checked when made, and actually used to do a restoration, as part of an annual training and retraining exercise). It is also important for there to be multiple levels of backup available, going back a good length of time, in case a software problem or configuration error was introduced but not identified for a period of time over which additional backup images have been made.

Complete system backup images used to require reels of magnetic tape and a lot of shelf space. Today, with BlueRay DVD media, high-capacity external USB-connected disk drives, and ultrahigh-density cartridge tape systems, a complete backup may fit in your coat pocket (which also makes them easier to filch). Making and verifying backups is a critical procedure and needs to be done properly and in a secure manner. History includes too many examples of organizations that either had no backup images or discovered that theirs were incomplete, out of date, or unreadable. A backup, which can be used for computer restoration, needs to be made (and verified) whenever you install patches or software updates. Backups don't always have to be total; you can make partial backups. But that means that full restoration may require loading a series of partials in the proper order, a process which just invites human error.

Since a backup image would afford a potential attacker with a wealth of information about your systems (they can be restored onto other systems and examined or virtualized), their transport, storage, and production all need to be managed in accordance with good physical security procedures. If you decide to use an off-site storage facility, it ought to also follow security procedures. (If it is too hard to steal backups from an operating facility, just get a job at the place where they are stored!)

For backup storage and retrieval, the storage facility should require your credentials (they might provide you with a key), and their personnel should not have access to your materials without your presence. Banks have figured out this procedure; look at their methodology for handling safe-deposit boxes. Actually, with backup now being possible on rugged, high-density (and physically small) media, such as recordable DVDs, it isn't inconceivable to use a bank safety deposit box for off-site storage. That is certainly better than not having an off-site backup cache, but the problem is that you still need an audit procedure and credential verification procedure (maybe requiring a call back to the SCADA facility to verify that backups have been requested), and most banks don't provide that additional level of security. If you are on the approved list and have suitable ID, you get the safety deposit box and privacy to do whatever you want with its contents. Off-site storage is considered as essential if there is the possibility of a facility being totally destroyed by flood, fire, earthquake. If these are unlikely possibilities, then another strategy is to have a second set of backups and store them elsewhere in the same facility.

Critical Ancillary Subsystems

For identification of the boundaries of your physical security perimeter, it is important to make sure that it extends to the subsystems that are essential to the operation of your SCADA system and facility. A SCADA system without communications to the field is useless. The same is true of one without electrical power. Environmental factors, like electrical power, telecommunications, and heating, ventilation, and air-conditioning (HVAC), are essential for a SCADA system to operate (and for operational personnel in the control room). In the same vein, the people who use and operate the SCADA system also need an acceptable environment, comprising HVAC, water, lighting, and sewer. An extremist group could probably shut down your SCADA operations (or at least force you to activate an alternative facility) by tossing a dead skunk (or a chemical agent) into the building air intake or breaking your water supply main. There are ways to eliminate this possibility (using chemical filters and positive-pressure rooms), but they tend to be costly and are much more easily done as part of an initial facility design, rather than as add-ons or as modification to an existing facility.

Fire suppression

Most modern facilities have some form of fire-detection and abatement system (usually smoke detectors and sprinklers). If the sprinkler system in the computer room isn't designed properly, shutting down and seriously damaging a computer system can be accomplished by setting off the sprinklers. The use of Halon as a fire suppressant had replaced most sprinkler systems in computer rooms, but the risk to human life—and to the ozone layer (Halon contains chlorofluorocarbons)—proved

to be too great. Today, a computer room (or control room) full of electronic equipment is best protected against fire by a Halon substitute; however, if this is not viable owing to local fire safety codes, then the sprinklers at least ought to be set up with independent zones, and the computer room(s) should be separate from other areas. It is also useful to have a programmed delay in sprinkler activation to permit cancellation of activation owing to a false alarm.

Telecommunications

Many SCADA systems previously used leased telephone lines provided by the local telecommunications service provider (in the past, these might be discrete analog phone lines, but today this might be a single T3 connecting the site to a Corporate WAN). Somewhere in the building where the SCADA system is located will be a *telecom room.* Too often, this room and the equipment it contains will be neglected when a security perimeter is defined, possibly because it is separate from the computer and control rooms or because the room is thought of as belonging to the telephone company. This is one place where an attacker could get access to all of your RTU communications (and if it is a digital circuit carrying TCP/IP traffic, then it would provide a pathway back to the SCADA system).

These days, this room may also house your routers, switches, and other LAN/WAN equipment. It is essential that this equipment be protected from physical access by anyone not properly trained and authorized. The telephone company might object, but it is in your best interest to ensure that even their personnel must be escorted by an authorized employee with the necessary credentials, to access these areas. For SCADA systems that use radio communications, the same need for physical protection applies to the master radio equipment, access to the cable from the SCADA system to the radio equipment, the cable from the transmitter to the antenna, and the antenna/mast itself (fig. 9–7). The easiest means of taking out your SCADA system might just be to to climb to the roof and knock down your antenna mast.

The attacker may realize that they can shut you down by going onto the unprotected roof of your building and taking down your communications equipment. All that stuff on your roof is like a bull's-eye painted on the side of the building

Fig. 9–7. Physical protection of roof areas with radio equipment

LANs

A potential attacker can also gain a lot of confidential information by sniffing out the Ethernet frames traversing your internal LAN (the business LAN and/or the SCADA LAN), as these could include internal emails, remote login sequences, file transfers, and SNMP traffic, for instance. Make sure that your LAN equipment (hubs, switches, firewalls, and routers) is inside your physical security perimeter. This should extend to the cabling used to connect the computer equipment to the switches (fig. 9–8). It has been mentioned that using the mechanisms available in Ethernet switches to create VLANs, set port security, and the message isolation provided by switched Ethernet, add a level of security against packet sniffing. This is true, but if an attacker can get physical access to the switches, it would be possible to connect to the console port and alter the configuration to create a SPAN port. Therefore, putting strong (and different) passwords on your switches is also a very important security measure. (And if you have them set to send Syslog messages, you will know if someone tries to login and fails, or tries to set/change a password). Physical port blockers are also a good idea for any unassigned switch ports.

WLANs really ought to never be part of a SCADA system because they are radio-based. But if you must use wireless Ethernet, even on the business LAN in the SCADA facility, physically secure the access points and the LAN wiring that connects them. In another chapter, we will discuss how to add cybersecurity to the WLAN.

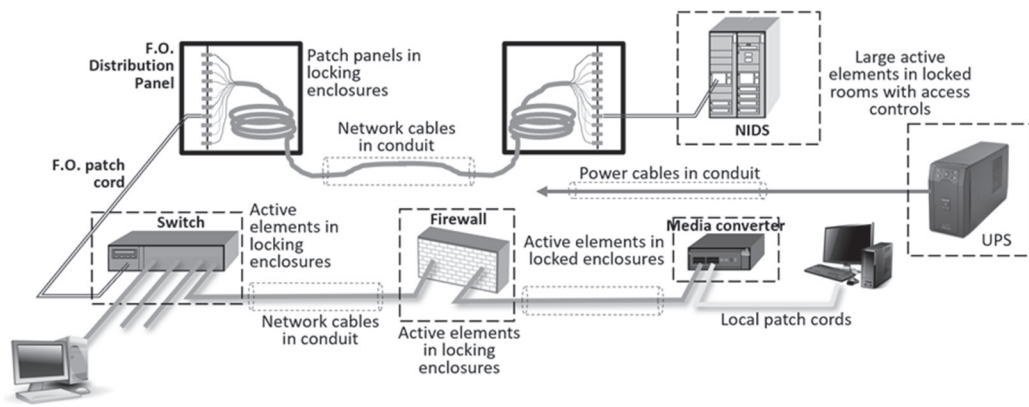


Fig. 9–8. Physical protection of network cabling and components

Electric power

Electric power is essential to run the SCADA system and the communications equipment, as well as the lighting, the HVAC, and even the access-control and monitoring subsystems. Cutting electrical power is an obvious way to take out a

SCADA system. Most facilities have a UPS or a standby generator or a combination of both. (The combined approach is best, owing to the time required in order to bring a standby generator up to operating levels once power has been lost.) Obviously, you ought to provide physical protection for the generation and UPS equipment. But you also need to think about the electrical distribution system and equipment, as well as the fuel supply for the standby generator (fig. 9–9). Eliminating the fuel source for the generator, prior to knocking out the power feed from the local utility, would be a good way to kill power to a facility.

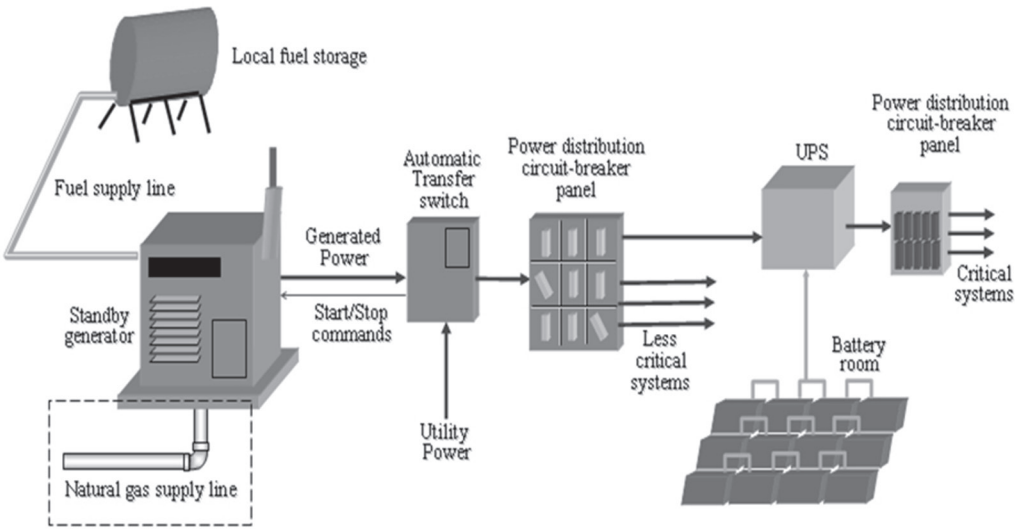


Fig. 9–9. Power supply: typical configuration and equipment interconnections

If the direct-current (DC) supply to the UPS is accessible, then it could be severed, disabling the UPS. It is also possible to damage the distribution and circuit breaker panels to shut off power to critical systems. The electrical infrastructure for your critical (sub)systems needs to be considered as a potential vulnerability and ought to be inside your physical security perimeter. In a public building, that can be an issue, as power distribution and circuit breakers and switches might be in a utility room or sub-basement area that is not within your control.

Remote and Field Sites

SCADA systems include the electronic equipment located in the field that is used to establish the I/O interface, to measure and control the distributed process and plant equipment. In most instances, this means some type of RTU or PLC device with serial or IP-based communications to the SCADA system. Otherwise,

depending on the industry segment and type of field site, this may mean a small DCS, a PLC control system, or a substation automation system.

It might appear trivial to consider the issue of physical security (of the SCADA system components) at a field site. For someone to get access to that equipment, they would already have gotten into the site and could just as easily smash things or manually operate the local process equipment. Field sites usually do have some level of physical access controls, even if just a chain-link fence with a padlocked gate. The more critical or essential the site is, the greater will be the protective measures employed. A large site such as a major pumping or compressor station might even be manned. Often intrusion sensors are wired as inputs to the local RTU in order to provide an access alarm in the SCADA control room. Normally, physical security measures around a field site will be implemented to keep out the curious, the vandals, and those looking for something to steal. Physical security is intended just as much to keep people from getting hurt or killed as it is to protect the plant and equipment. A determined and skilled attacker would find it reasonably easy to break into most unmanned field sites. If you can't really secure a field site, at least you can attempt to monitor it so you can respond to a physical incursion. That usually means having sensors (motion, door-open, infrared, sound, etc.) tied into the RTU as additional contact inputs, setting alarms on those inputs, and adding them to the operational displays. If there is broadband IP networking to a field site, then you may also have the option of having video monitoring at the site, possibly with motion detection capabilities. You may also have the option of installing a commercial burglar alarm system from a monitoring company (e.g., ADT, AT&T, etc.) and letting them monitor and notify you (and local law enforcement) if an intrusion is detected.

It has already been mentioned that in many cases, leased telephone lines (or digital communication circuits) are often terminated outside the facility by the communications supplier and are then routed into the site. This holds true for electrical substations, in particular. Such a configuration makes it possible for an attacker to gain physical access to the telecommunication circuit without having to break into the site. An attacker with a laptop PC, a modem (or the digital equivalent), and a protocol test set could then take control of the communications link and send commands to the equipment at that field site. If the circuit is a multi-dropped circuit and the attacker selects the first drop site, then he or she would be able to send commands to the field equipment at all of the other sites that share the multi-dropped circuit. The damage that can be inflicted by taking over one or a few field sites depends on the nature of the site and of the process. Tripping a particular circuit breaker, taking out a selected transformer, closing a particular valve, or starting a selected pump can have minor or major impacts. The use of serial link encrypting devices, VPN technology and/or firewalls would eliminate this threat as long as the attacker could not gain building entry.

IP networking in the field

Some SCADA system operators have run TCP/IP networking to their field sites. This is happening in part because of the disappearance of traditional analog telephone service and in part because IP networking offers a lot of additional functionality and can be implemented with commercially available hardware and software. The same telephone line that could only support 1,200 bps analog communications can be replaced with a DS0 digital line that supports 64 kbps data transmission or a fiber optic cable that can support Gigabit speeds. A single IP-network circuit can be used for voice and data and can support multiple, concurrent traffic streams.

Personnel who go to field sites could, in theory (though limited by available bandwidth), be linked into the corporate email system, access files, drawings, and manuals and even be bridged into the corporate phone system and the Internet. This may sound great, but the problem is that this also means that a remote field site could, if not adequately protected, be the point of access for an attacker who wants to get into your SCADA system. Breaking into a remote field site is probably a lot easier than breaking into corporate headquarters and the SCADA control room. And with IP networking, making an unauthorized network connection at one field site could, in theory, provide direct network access to both the SCADA system and all of the other field sites and interconnected systems, PLCs, and RTUs. This is one reason for considering using local firewalls at each site and between the field and the SCADA system. We will discuss specialized industrial firewalls in Chapter 11. It would also be good to create a VPN between the central SCADA facility and the field sites, but this won't protect against an intruder who gains site access, as the VPN would only extend to the router at the site. It would not, for example, prevent the attacker from trying to send control commands to other sites. So, at the field sites, it would be advisable to place critical equipment and interconnecting wiring in locking enclosures (with open-door alarm switches) for added physical security. We will discuss cybersecurity measures elsewhere; this chapter is focused on physical security measures.

Adequate physical security provides a solid base on which to layer adequate operational and cybersecurity. In the next chapter, we will discuss operational security.

10

Operational security

Policies and Administrative Controls

As has been mentioned, maintaining the security of your critical systems, information, and infrastructure is not based totally on the application of technical remedies, such as video cameras, firewalls, and intrusion-detection systems. Security begins with the process of establishing a corporate culture that identifies security as a critical goal and that provides incentives to encourage employee acceptance and active participation in achieving that goal. It is essential that security not be perceived as some bureaucratic program that management has forced on employees for the sake of avoiding legal hassles, but as something critical to the success and well-being of the organization. Security needs to be a top-down objective, supported at the highest levels of the organization. If top management doesn't take security seriously, then neither will middle management or the rest of the employees (if a senior manager refuses to occasionally change their PC password, then employees will question why they are required to do so). They may give it lip service, but they won't really adopt it as a necessary and critical aspect of the way they perform their duties. And it is likely that some important cybersecurity-related processes and procedures will be skipped, if inconvenient.

Real security requires a proper mind-set: the sincere acceptance of the need for security-oriented habits and procedures. For a SCADA system operation—particularly one that supervises and controls a critical, essential, or dangerous process—proper operational, physical, and cybersecurity could be a matter of life and death.

The starting point for security is the establishment of written policies that define the organization's goals, intentions, and rules with regard to security. Written policies are crafted to address the business requirements of the organization, as well as to address the legal and regulatory issues facing the organization. The challenge is to have well-considered policies that don't go overboard, while also limiting the number of policies, so as not to create a policy manual that looks like the telephone book for New York City (well, back when such things were actually printed on physical paper).

It is also important to understand the difference between a policy and a procedure. A policy states organizational beliefs and objectives, whereas a procedure defines how those objectives will be implemented. Normally, every policy will have

one or more procedures that are used to implement the policy. (An overarching password policy would have specific procedures for dealing with different operating systems and with smart devices with password limitations and equipment where a shared pass/PIN code is all that is possible.) Management must lead in setting policy. However, it is usually best to have active employee participation in developing, testing, and revising the procedures that carry out the policies. For the purposes of overall security, including cybersecurity, there need to be written, enforceable policies that address issues such as

- Protection of critical and confidential information
- Protection of critical cyber assets
- Use of corporate computing and networking resources
- Texting, email, Web browsing, and general Internet access
- Use of cell phones, tablets, and other mobile devices
- Restricting cell phone capabilities such as camera, video, and hot-spot support
- Use of Wi-Fi and other wireless technologies
- Discarding/repurposing of computer equipment
- Discarding/destruction of confidential materials
- Information access control and categorization
- Management's role and responsibility for security
- Personnel training and security familiarization
- Personnel hiring and termination practices
- Physical access control and monitoring
- Use of removable electronic media
- Systems management and administration
- Personnel physical and electronic surveillance
- Software updating, testing, and patching
- SCADA configuration management
- Application development, testing, and maintenance
- Third-party application software use and maintenance
- Supply chain protection and monitoring

This list is by no means intended to be comprehensive, but it is a good starting point for policy definition. Policies don't have to be especially verbose. Actually, a clear, concise policy is more likely to be understood and followed than one tortuously crafted by a team of lawyers (although your legal and human resources staffs should review and approve your policies). For example, a policy regarding adding software to a company computer could be as simple as the following:

> *All computers owned by the company and used by employees in the execution of their assigned duties will be configured with company-approved operating*

system and application software, which will be installed and maintained by the company IT department. No employee shall add, remove, or modify the software on their company-supplied computer, including downloading of software from the Internet or installing software owned by the employee. Employees who believe they require additional or different software to accomplish their assigned duties will contact the IT department and define their requirements. Only the IT department may modify the approved set of software on individual computers. Employees must use only company provided computers and software in the execution of their assigned duties.

Policies, particularly those aimed at cybersecurity, need a periodic review and updating to match the current threat and technological environment (e.g., a policy regarding use of floppy disks should probably be replaced with one regarding [re] writable CDs or DVDs and Universal Serial Bus [USB] *flash drives*). For the electric utility industry, the NERC recommendations for cybersecurity state that someone in upper management must be assigned to oversee and enforce the policies; that there be written procedures governing authorization of exceptions/deviations to any policy, including justifications for the exception; that there be an audit trail of such exceptions, as well as of any modifications to any policy; and that there be an annual review and evaluation of the policies and related procedures.

Although government regulations change over time, every organization currently must be concerned also about cybersecurity issues related to federal and state regulations and compliance with any associated requirements (and the NERC CIP standards or 10 CFR 73.54, if you are an electric utility). Thus, these regulations must be considered when establishing policies. Although the types of business and personal information specifically targeted for protection by SOX and HIPPA are unlikely to be stored on a SCADA system, the interconnection of the SCADA system to the business/corporate LAN/WAN makes a possible access path to that data—a flaw that must be addressed to be in compliance with the requirements of those laws. Although policies ought to be customized to meet your particular business and operational goals and objectives, there are numerous sources for prewritten, example policies that can provide a good starting point. (A Google search will yield a number of potential sources of example policies.)

Policies, no matter how well-crafted, are meaningless unless known and understood by the employees and backed up by consequences. Employee understanding ought to go beyond rote memorization; employees need to understand the threats and consequences of not having the policies in place and enforced. When people understand the reasons for a policy, they tend to adopt and support it, which is particularly vital when many of the policies (and resultant procedures) may seem to introduce time-consuming and counterproductive requirements that add to an employee's workload.

Procedures

Once policies are established (written and formalized), procedures that implement the policies must be crafted. These procedures need to be carefully considered, because long and excessively complicated or laborious procedures tend to be ignored or bypassed. A procedure is like cooking with a recipe; it contains a clear set of steps and directions for executing a process, so that the process is done properly and so that no critical step is forgotten or done out of order. Properly crafted and validated procedures enhance safety, security and greatly reduce issues of human error. Participation from employees is essential in development and testing of procedures, since they are the people who will have to follow the procedures. Personnel are far more accepting of procedures if they have a hand in developing and validating them. Procedures need to be reviewed on a routine basis—to improve, streamline, and correct them, as well as to keep them current. Similarly, procedures (and policies) should be retired when they no longer serve a purpose.

One reason for the development of good procedures is to capture, encapsulate, and document the expertise and experience of the personnel involved. Procedures can be written at many differing levels of detail, depending on who is to follow them. If everyone following a given procedure will have an advanced degree in mathematics, then a procedural step could merely state "Compute the standard deviation of the data set." If this is not the case, you will have to put in the details of how to collect the necessary data and make that statistical calculation. Well-developed and documented procedures can double as training materials for new employees. Emergency procedures that are (hopefully) not required on a routine basis—such as restoring the SCADA system from the backup media—need to be especially clear, detailed, and complete AND these procedures need to be validated through testing. Procedures that are rarely, if ever, performed need to be the most clearly and accurately documented since the personnel performing them may never have done so previously.

Procedural validation

Procedural validation is merely walking through the procedure, like a dress rehearsal for a play, to insure that it actually includes every necessary step, in the correct order, plus any potential variations that might be necessary—and to insure that all required materials and resources are actually available and functional (this process is also often called a *table top review*). If the procedure requires using backup CDs that are locked up in a secure, fireproof cabinet and obtaining the cabinet key from an authorized individual isn't part of the procedure, then the procedure is a failure. If the procedure requires making a voltage check and the necessary test equipment isn't available to the personnel who will be performing the procedure, then the procedure is a failure. It is generally a good idea for someone not involved in developing a procedure to be selected to perform procedural

validation. Excessive familiarity with a procedure and process can cause the procedure writers to miss a critical step that needs to be documented because they are so familiar with the process. Because they are so familiar with the process, they automatically perform that step without thinking, and thus they don't put it in the document (like telling someone to hit the "Enter" key after typing a command on the keyboard). Again, the level of detail needed in a procedure will depend on the frequency of its use and the training, experience, and expertise level of the personnel who follow the procedure.

Critical procedure sets

It is worth enumerating here some of the procedure sets that are critical for SCADA system security, as well as the issues that each addresse:

- **Employment initiation and termination.** This includes procedures regarding the requirement of drug tests and financial and police background checks on prospective employees, as well as periodic rechecks on employees with high access levels; procedures for having employees sign nondisclosure agreements (NDAs); and procedures for collecting company property, canceling access rights and credentials, and reviewing ongoing legal obligations under the NDA during exit interviews.
- **Contractor supervision and management.** This is similar to the previous procedure set, except dealing with contractors. This includes procedures for verifying that a contractor organization meets (or exceeds) the minimum acceptable security standards of your organization (e.g., that the contractor organization also does background checks and drug testing).
- **Credentials, password, and account management.** This includes issuance and cancellation of accounts and credentials, upgrading and downgrading of access right as duties change; procedures for dealing with lost and stolen credentials; and personnel password guidelines.
- **Security-patch/virus-scanning management.** This includes installing and testing security patches, distribution of patched software, and making backups of new versions, as well as procedures for routine virus scanning and updating of virus-scanning software. In fact, any technical countermeasure that uses signatures, such as NIDS and AV scanning, will require periodic updating.
- **Software-update validation and testing.** This is the methodology used to update, track, and test operating system, networking, application, and third-party software in the offline environment, prior to release to production system.
- **Installation of modifications to the production system(s).** This establishes who is to perform the work, as well as what sequence is

to be followed for redundant and alternative site configurations. This also includes procedures for reverting to the prior version (*rollback*) if problems arise during the modification process. Operational conditions under which this is or is not to occur are outlined, with attention to required coordination and approval from all necessary parties.

- **SCADA system startup and shutdown.** This is probably provided by the SCADA system vendor and includes normal steps taken and sequence followed to gracefully bring the system up or down—primarily for use before and after hardware replacements, repairs, or software upgrades.
- **SCADA system restoration.** This is used when the system has crashed or been otherwise disabled. Assumptions regarding possible hardware failures and/or software corruption are outlined. Procedures include performing necessary visual inspection and unit tests of equipment, replacement of damaged equipment, power-up sequence, required restoration media and correct version, overwriting of damaged/infected software, and proper sequence of software installation, as well as restoration of configuration, restarting the various servers and other computers, restoration and verification of RTU polling, initial alarming override, restarting operator functions, restoring links to other systems, and so forth.
- **Employee security training.** This addresses the types of training needed for given levels of access, the frequency at which retraining will be required, and metrics for determining training efficacy. There should be basic cybersecurity familiarization training, including a discussion of social engineering, for all personnel, and it ought to be refreshed periodically. Personnel directly responsible for supporting and maintaining technical countermeasures will need to have more advanced training, including product-specific, vendor-provided training.
- **Electronic security incident response and reporting.** This includes possible symptoms or indications of attack, actions to take and their order and priority based on incident type, personnel who need to be informed, and reporting requirements. A basic knowledge of computer forensics and information gathering and preservation will be essential for personnel that deal with incident response.
- **Physical security incident response and reporting.** This includes personnel safety issues, facility and equipment safety issues, actions to take and their order and priority based on incident type, personnel who need to be informed, and reporting requirements. If your organization is subject to a regulatory or state/federal agency, then the process must, of course, incorporate their reporting requirements.
- **Firewall, AV, SIEM, NIDS, and HIDS updating.** This outlines who is authorized, reasons for updating, periodicity of updating, processes used to ensure security during updating, audit log of updates, and

validation and testing of updates. The process for obtaining updates and transferring them to these subsystems must ensure that malware is not propagated via the process.

- **Sensitive/confidential material management.** This includes identification and classification of materials, methodologies for physical protection and backup, and mechanisms for access control, duplication control, and modification control, as well as procedures for storage and destruction of materials.
- **SCADA system backup generation and storage.** This addresses events that should trigger a backup, how many prior backups to maintain, who can make a backup, on-site and off-site storage, access to on-site and off-site backups, remote storage access procedures, authority level for access, and approvals for access, as well as validation of backups and creation of an audit trail of backups.
- **Communications subsystem restoration.** This includes contacting the telecommunications supplier to report problems, checking the status of and restarting the telecommunications and networking equipment, as well as understanding equipment diagnostic indicators.
- **Electrical power restoration.** This outlines how to manually (re)start the backup generator, how to enable or bypass the UPS, and how to configure the necessary switch gear and circuit breakers.
- **Access to secure areas.** This establishes the access-level authority required for each category of restricted area, requirements for entry and exit, requirements for bringing materials in or out of a restricted area, and procedures for accompanying quasi-authorized personnel (people who occasionally have a need for physical access such as a vendor maintenance or facility maintenance employee) into restricted areas.
- **Remote electronic access.** This includes setting up electronic credentials, establishing a VPN connection, strong-authentication procedures (issuance and revocation of digital certificates), dial-in telephone mechanisms (if applicable), corporate WAN mechanisms, and Internet mechanisms. It should also address the periodic re-issuance of credentials and changing of passwords.
- **Collection and retention of logs and audit files.** This addresses frequency of collection, verification of logging, storage duration, storage media, and their handling and physical protection. If a SIEM is implemented, then this can be part of SIEM maintenance procedure. Part of this procedure is to verify that all devices that ought to be forwarding logs are doing so and are configured properly to generate logs of adequate detail.
- **Addition and retirement of computing equipment.** This includes installation of baseline operating system and application software;

establishment of network settings, security policy settings, and user accounts; installation of virus-scanning and firewall (or endpoint protection) software and settings; installation of HIDS or other endpoint protection, software and electronic-access monitoring and surveillance software; and disposition, destruction, and/or disabling of key components.

- **Fire alarm response and reporting.** This includes evacuation of the area, verification of the validity of the alarm, personnel safety, notification of and coordination with police/fire departments, and shutoff of gas and electric service. If key equipment needs to be shut down or critical materials moved to safety, this should be addressed as well, including authorizations needed, who must be contacted, etc.
- **Social engineering incident response and reporting.** This outlines official response to a perceived social engineering attempt, telephone and email guidelines for social engineering ploys, requirements for request verification, and necessary credentials and credential validation.

Undoubtedly this seems like a lot of procedures (and it isn't even a comprehensive list), but keep in mind that your personnel are inventing informal, undocumented, and unofficial procedures all the time. They must do this in order to do their jobs, if you have not provided a formal, approved procedure. The problem is that two employees, with similar responsibilities and duties, will generally come up with two different (though probably successful) procedures to accomplish the same task. By formalizing procedure development, you can apply best practices and end up with a standardized yet optimized methodology that can be passed on to new employees. With proper periodic reviews, the procedures can be improved and modified to meet current environmental (business, legal, and regulatory) demands.

Operational Differences

Supporting a SCADA operation is different from supporting a corporate IT organization, an application/Web server farm, or a software development environment. For that reason, some of the policies and procedures that would be typical in those "IT" environments are not typical (or even practical) in a SCADA system operating environment (e.g., you won't find a help desk, and rarely will there be any policy regarding performance and capacity management). All of the aforementioned operations, including SCADA, involve a lot of computers and networks and software. However, the main purpose of a SCADA system is to monitor and control a well-defined, infrequently (if ever) changing process and equipment set (the process itself, not the condition/state of the process), so there is little motivation to make some of the changes that would be typical in other environments.

Once a SCADA system is operational and fully debugged, most owners/-operators are loath to make even slight changes, owing to the all-too-true possibility of breaking something as a result. In other environments, it would be commonplace to keep operating system, networking, Web server, and commercial application software up to the latest and greatest versions—and not because of security alone (although that is a *big* part of it today). If a Web site doesn't support the latest features of PHP scripting, Perl, or JavaScript, or if a server farm isn't running the latest update of Microsoft Office 365, then they are not fulfilling their mission. Also, their procedures for installing updates are generally in the form of "load and run the service pack." For a SCADA system operator, there has to be a very good reason to allow software, configuration, equipment, or application changes. The only reasons for changing software that are usually considered acceptable are because there is an intolerable bug in the software and/or because the vendor is dropping its support of the old version of the software or hardware (and maybe not even then).

For big SCADA systems, supervising critical processes, the update process must involve trying changes out on an offline (non-production) system first or, at a minimum, on the backup system. Today, as SCADA systems are linked into LANs and WANs and exchange information with other systems, and as corporate IT groups are called on to supply technical support for the computer portions of these SCADA systems, there can be a conflict between the perceived mission of the IT group and that of the SCADA/operations group. Clearly written policies and procedures that are jointly crafted by these two groups represent a way to bring these two cultures together while addressing the unique requirements of both. And, like it or not, SCADA systems today need security patches every bit as much as (or even more than) an IT server farm does. On the other hand, the IT group needs to understand the operational differences that are unique to a SCADA environment. It may be undesirable, and have a revenue impact, to take down an e-commerce Web site for an hour to install patches and upgrades, but it is potentially life-threatening to do so with a SCADA system. Worse still, restarting a SCADA system is not just a matter of rebooting the computers (which may solve a lot of conventional IT problems, but it can be a last resort to a SCADA operation). SCADA systems control physical processes and, in order not to send out dangerous or damaging control commands, a SCADA system needs to scan the field devices and determine their status and configuration, and that of the process, in order to be ready to support supervisory control and alarming functions. If RTUs need to be reinitialized (or even reloaded with configuration information and control logic), that adds another level of complexity. During the electrical blackout in the northeastern United States back in the 1990s, the process of bringing the various regional SCADA/EMS systems back online and piecing the electrical grid together again took several days of continuous work. Bringing a long-haul product pipeline back into operation is similarly complicated. IT personnel providing support to SCADA system organizations need to propose all activities and actions in advance,

have them discussed with the operational team, and then perform and coordinate their work using mutually developed procedures. On the other hand, it is no longer realistic for a SCADA system owner/operator to take the position that security-related patches and updates can be ignored.

Training

Your own personnel are normally the first line of defense against certain types of security threats, particularly the current or former hostile insider and an attacker attempting to gather information (or gain unauthorized access) by using social engineering tricks. (Pretending to be an ex-employee returning, just for a moment, to pick up some forgotten personal effect is a typical social engineering ploy.) Nevertheless, for your personnel to be effective, they must not be denied appropriate and applicable training. Certainly, part of that training includes being familiarized with company security policies and learning the procedures relevant to all employees (e.g., fire safety) and those procedures associated with their particular duties. (For example, you wouldn't teach every employee how to make a SCADA system backup, because not every employee will be authorized to perform that task, but you should consider some level of cross-training so that there is always someone available who knows how to perform any critical, essential operation). Part of training is to educate personnel about how seriously the company values security and how important it is to everyone who is part of the organization.

It is highly recommended that all employees in an organization that owns and operates a SCADA system take part in an initial security-training/familiarization seminar immediately on starting employment, just as a new employee at a chemical plant or refinery would attend a plant-safety seminar. The following are key points that all employees need to know:

- Awareness that cyberthreats do exist and awareness of the potential threat sources
- Awareness of the means used by attackers to plan and stage attacks and to gather confidential information (malware, hacking, social engineering, etc.)
- Awareness of basic practices that support and enable an effective security policy (passwords, access control, information security, scanning and using only authorized portable media, etc.)
- Awareness of the potential damage/impact, both to the organization and to the outside world, of a successful cyberattack
- Awareness of management support for the necessary policies and procedures

One problem with security training is that it becomes outdated as technology and hacking methods evolve. Thus, there needs to be a mechanism for updating people's training and for keeping people apprised of new threats. One problem

with security awareness is that, like any other skill or knowledge that isn't used or applied on a regular basis, it fades over time. Consider the threat-alert color scheme adopted by the DHS to advise the population—and the appropriate military, civil defense, and law enforcement agencies—of the terrorism threat level. Because nothing noteworthy has happened in the United States since 9/11 (even though the public would not be aware of attacks that have been *prevented*), many people now basically ignore the color system and any announcement of an increase in the threat level. (Do you actually realize that there is a color code system and that we have been at a high alert level for nearly 20 years, as of this point in time?)

Keeping people interested and focused on security is not an easy task, given all that they normally need to deal with in their jobs. It is kind of like safety training; the day after someone is seriously injured or killed in a plant, everyone suddenly takes the time to follow the recommended safety practices, even though the prior day they were bypassing or rushing through some of them. There is no single solution to this problem, but different organizations have used the following strategies to keep security in the forefront of employees' attention:

- Annual or bi-annual employee meetings, to reaffirm the need for security
- Metrics that track compliance, offering rewards for the best performance
- Annual retraining/training-update seminars
- Bonuses to managers who meet security objectives
- E-newsletters that provide information on new threats and actual events
- Security certification program, offering rewards at various levels (up to black belt) of certification
- Computer penetration testing (pen testing), sharing the results with staff
- Physical penetration testing, where the attacker leaves visible indicators in areas successfully penetrated
- Internal simulated phishing attacks to keep people on their toes

Recovery Procedures

As has been previously mentioned, well-documented and validated system recovery procedures are essential to a SCADA system operation. Most SCADA systems, once commissioned and stable, tend to operate 24/7 for years without mishap. This is obviously a good thing, but it does mean that personnel responsible for performing a system restoration, if one is ever required, have little or no opportunity to practice their duties. This makes it even more important that the procedure be very clear, well-documented, and fully validated. Unfortunately, full and complete validation of the restoration procedures may not be possible, because no one wants to bring the system back down, once it is fully operational, for procedure validation purposes.

If there is a *test and development system* available, separate from the production system, then this might be usable for procedure testing. However, it is common for such test and development systems to include a minimum set of hardware and software, usually in a nonredundant configuration. Thus, such a system may only approximate the true system architecture and would thus limit the ability to do a complete procedural validation.

Developing well-documented and validated procedures may require the assistance of the SCADA vendor, who normally has systems on their factory floor (if they offer integration services or turn-key solutions) or training facilities that can be used for this purpose. If the SCADA system is fully redundant, then it may be possible to do a good bit of the validation (and training) on the standby equipment. If the system is supported by an alternative facility, then the SCADA system at that alternative facility can be used for procedure validation and training. Vendors can usually provide generic training (and maybe documentation) on system recovery and restoration, but the site/customer-specific and application-specific portions of the procedure will still have to be documented and added into the recovery procedure by the SCADA system owner. Even if a full procedural test cannot be performed, due to the lack of backup or alternate facilities, it is possible to do a *structured walk-through* or a simulated execution of the procedures. In these cases, the knowledgeable experts in each area of the procedure follow along as the testing/training personnel follow the steps of the procedure (without actually doing the disruptive things like turning equipment on or off, loading software, etc.)—with the experts verifying that the individual steps and actions, as described in the procedures, are complete, in the right sequence, and technically accurate. A structured walk-through is a good way to test a procedure, prior to its issuance, as well as to train personnel in the use of the procedure, once formally issued. There was an actual case of a SCADA system owner/operator who had updated some system equipment to eliminate legacy magnetic cartridge tape drives and replace them with Blu-ray DVD technology, but whose system backup was still on magnetic tape cartridges (this was discovered when they did a tabletop review of their restoration procedure).

Annual Review

Various governmental agencies and industry standards organizations recommend that critical procedures be practiced, and policies and procedures be reviewed, on at least an annual basis. The objective here is to ensure both that personnel remain competent and capable of performing these procedures and that the procedures remain accurate and up to date. If there is no reason to make significant revisions to a procedure (e.g., the SCADA system underwent a major upgrade or was replaced), then total procedure revalidation generally isn't necessary.

The annual rehearsal (for retraining purposes) of critical procedures can be performed on one of several levels. The procedures can actually be performed on backup equipment; they can be simulated; they can be subjected to a structured walkthrough; or they can just be given a checklist review. On the one hand, the latter alternatives are less involved and nondisruptive, but don't provide the same training experience as actually performing a procedure or at least simulating its performance. On the other hand, a structured walkthrough or checklist review could be performed by non-experts and could be audited by those experts who developed the procedures; thus, these review levels afford an opportunity for expert oversight (and revision, if needed) of the procedures.

Background Checks

As objectionable as it may seem to inquire into someone's private life as a requirement of employment, it makes sense that people placed in positions of trust ought to be trustworthy. Today, if just to avoid the legal liabilities, most companies now require pre-employment drug testing and random drug and alcohol tests for all current employees (and possibly for contractors performing sensitive or vital activities). Government positions dealing with secret matters require a security clearance that necessitates a detailed background check. Thus, if a person is going to be allowed to operate, program, or maintain a SCADA system, it makes sense that they be willing to accept some level of background check, both as a requirement of employment and on an annual basis. (Of course, a company should also remember to impose a drug and alcohol policy on these individuals.)

The recommended policy for people with top-level access (and this could include contractors) is to also have criminal and financial checks performed. The reason(s) for doing a criminal check ought to be obvious, but why require a financial check? Insiders in various government and private organizations have been compromised by financial motivation since the beginning of civilization. A person who has gotten into financial trouble previously—or is just plain greedy—would be a perfect target for an external threat agent seeking the means for staging an attack. Whether by giving out an ID and password or providing a set of access-control credentials, people have been known to "join the dark side" for the right amount of money. By reviewing the overall financial history of an employee, it is often possible to distinguish those who might be good candidates for recruitment (and for company-provided financial counseling) from those who aren't, and even to identify those who have inexplicable income. If an employee is aware that financial and criminal checks will be made on an annual basis, that provides added incentive to avoid temptation. Of course, personal financial and criminal information needs to be kept confidential and treated with proper discretion, as should any other personal information obtained and retained by the company. It is important that policies and procedures related to background checks be reviewed

by your legal and human resources departments, for compliance with laws and regulations.

Now that we have discussed physical security and operational (administrative) security, our next chapter will address cybersecurity—specifically, technical measures that can be used to achieve acceptable cybersecurity.

11

Computer systems & Network security

In the same way that you need to establish and maintain a security perimeter that physically protects and isolates your personnel, critical systems, and sensitive and confidential materials from threat agents, you need to establish and maintain an electronic security perimeter that protects and isolates your systems, software, and electronically stored data and confidential information from cyberattacks by threat agents. As with physical security, you need to identify and limit (or eliminate) attack pathways through this electronic perimeter. A combination of defensive mechanisms, detection mechanisms, and architectural strategies can be combined to provide an adequate level of cybersecurity. Although your SCADA system as a whole needs protection, there will be countermeasures that should be applied on each individual computer (PCs, servers, and even VMs) and countermeasures that will be applied to your networks and active network elements. It would be nice to believe that you can create an absolutely impervious set of cyber defenses, but experience has shown that the bad guys eventually find a way to breach them. So, detection, impact mitigation, response, and recovery strategies (and countermeasures that support them) also need to be part of your cybersecurity program and plan.

In prior chapters, we have discussed the importance of administrative security activities such as account management on all of your computers and smart devices, ensuring that you have backups and everything else you need so you can restore all or part of your SCADA system, and having procedures and processes to ensure that portable media and digital devices can't be used as an attack pathway. So, we will not be repeating those discussions in this chapter, but they are just as essential as buying the latest and greatest technical countermeasures.

Having said that, there are, of course, a number of technological countermeasures that can be applied to create or enhance cybersecurity, and in this chapter, we will discuss as many of them as possible, including their strengths and weaknesses. These technologies include the following:

- Firewalls (Enterprise, Next-Gen, Industrial)
- Network Intrusion Detection (and Prevention)—NIDS
- Host Intrusion Detection (and Prevention)—HIDS
- Whitelisting and Blacklisting
- Endpoint Security

- Malware (AV) Scanning (signature, heuristic)
- Configuration Monitoring
- Network Status Monitoring
- Smart Switch Security (ACLs, MACsec, port security, SB-NAC)
- Data Diodes
- Syslog/Log Analysis
- Security Information Event Management (SIEM)
- Virtual Private Networks (VPN)
- Virtual LANs (VLAN)
- Device and system 'Hardening'
- DMZ Isolation
- KVM Switches

As we are talking specifically about SCADA system cybersecurity, and not general IT cybersecurity, we will not be addressing things that are specific to Internet-related cybersecurity, such as how to secure email, how to protect your Web server, and how to protect your DNS server from compromise (or your DNS clients from being spoofed). Your SCADA system should not be connected to the Internet and should not have a pathway to the Internet such that Web browsing and email would even be possible from the SCADA system. And if for some reason you do need to know about these issues, then your IT department should have the necessary expertise, as that is what they usually deal with on a daily basis. We are going to focus on how to apply technical countermeasures and architectural strategies to establish a strong level of cyber security for your SCADA system.

Firewalls

A good starting point for defensive technology are those devices called firewalls or security appliances. Classically, a physical firewall was a barrier that would prevent the spread of fire from one room to an adjacent room. Similarly, in cyber terms, a firewall keeps something malicious and undesirable on one network or LAN segment from spreading into an adjacent network or LAN segment. Firewalls are actually software, but many vendors sell that software already installed on a specialized, high-performance computing platform. There is firewall software, though, that you can purchase (or download for free) and run on any computer (a so-called host-based firewall). "Personal" firewalls are a variation on that technology, and both Windows and Linux come with firewall software (e.g., Windows Defender™ and iptables or ClearOS for example). Firewalls range in capabilities and functionality from an "Enterprise" firewall (also a "Next-Gen" or UTM Firewall), which would be something placed between a corporate network and the Internet, down to simple stateful packet-inspection/filter firewalls. There are also specialized

firewall variations, such as a proxy-server firewall or a deep packet-inspection firewall, and today, there are even industrial firewalls that are *protocol-aware* (down to the protocol function code/command and data object level) of commonly used Ethernet-TCP/IP industrial (application-layer) protocols such as Modbus/IP, EtherNet/IP (CIP), ProfiNET, DNP3 and OPC. We have already discussed the importance of using firewalls within your LANs, to segment them and restrict and control traffic flows—to prevent a compromise in one system element from then enabling unrestricted access to attack all other system elements (e.g., preventing the compromise of the primary SCADA server [or primary operating site] from then easily jumping to the backup SCADA server [or backup operating site]).

Enterprise firewalls tend to be feature-rich and must support significant throughput (packets per second) even with SSL message decryption taking place, because all of your traffic in from and out to the Internet passes through this firewall. These firewalls tend to support VPN gateway functionality, perform NAT services, web site black listing and usually perform malware detection and blocking functions. The biggest problem is usually that you can't turn on all of the available optional functions and still maintain a high level of throughput. It is not uncommon to see multiple enterprise firewalls used with a load balancer to gain the needed throughput. But, since the SCADA system should not be connected to the Internet—and the average message traffic on a SCADA system will be nothing like what occurs at the connection of the corporate network to the Internet—it is unlikely that you will need an Enterprise firewall for the SCADA system (but your IT department should have placed one between the corporate network and the Internet). Enterprise firewall software can be purchased (e.g., from Checkpoint) and run on a hardened Windows or Linux operating system platform. You can even outsource an Enterprise firewall function as a cloud-based service, so support and maintenance (including signatures and rules) will be managed by others (but of course that only works if you are connected to the Internet). Enterprise firewalls generally require extensive configuration, including selecting and loading applicable rule sets, based on the services and protocols you use—some provided by the vendor, as well as site/corporate-specific rule sets that you craft.

Most Enterprise firewalls have now been re-branded as Next-Generation Firewalls (Next-Gen or NGFW), which does require that they support important features and capabilities. An important Next-Gen capability is to be able to identify application traffic, regardless of what port is being used (i.e., not using the well-known port number formally assigned to that service/protocol) and to perform SSL decryption to ensure that content is visible and available for analysis. Several collaborative services and video conferencing services play fast and loose with port numbers and use SSL-encrypted connections to try and hide content. Hackers try to use the same schemes to create covert command & control channels to malware they have injected into your systems. The key to NGFWs is the ability to do everything a traditional Enterprise firewall does with the advanced

capabilities that address the "tricks" used by Enterprise (Net) 2.0 applications. Again, it is unlikely a Next-Generation firewall, whose mission is to protect against advanced Internet-based threats, would be appropriate or applicable in your SCADA system (assuming it has no pathway to the Internet).

A Unified Threat Management (UTM) firewall is similar to an Enterprise firewall but usually offers only a subset of their capabilities. Unified threat management appliances, also known as multi-function firewalls, have evolved from traditional firewall and VPN appliances into products that have many additional capabilities, such as URL filtering, spam blocking, spyware protection, intrusion prevention, gateway antivirus, and providing a centralized management, monitoring, and logging function. Traditionally, these functions were handled by multiple systems. UTM appliances generally come pre-configured and require far less user configuration setup than do Enterprise firewalls. UTM appliances are supposed to block a broad range of network threats before they have the opportunity to enter your network, by performing analysis (and not just signature-based) on the message streams:

- Protocol Anomaly Detection (SPA)—Internet standards for data traffic are enforced to detect and block non-conforming traffic and isolate threats (a.k.a. Stateful Protocol Analysis).
- Behavioral Analysis—Hosts exhibiting suspicious behaviors (i.e., scanning) are identified, and potential denial of service attacks can be blocked.
- Content Blocking—High-risk file types (e.g., "exe"), known to propagate viruses or attacks, are flagged and deleted before they enter your network.

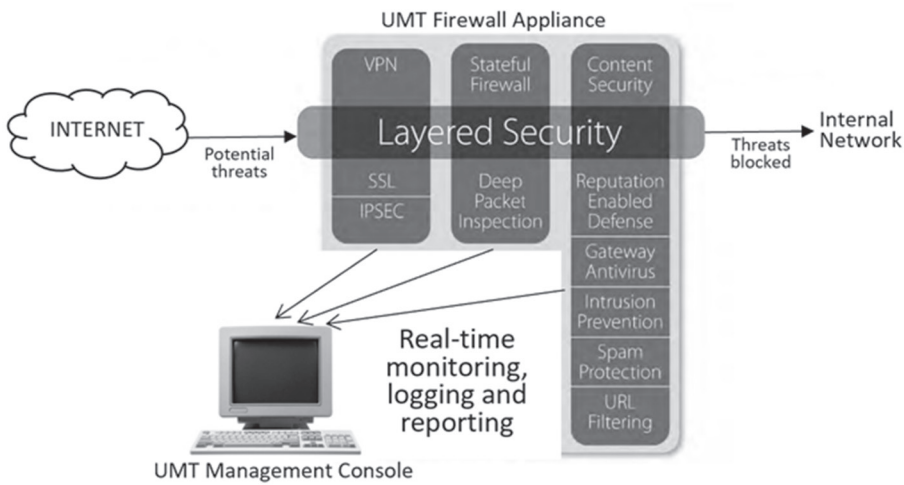


Fig. 11–1. Unified Threat Management appliance

It would not be unreasonable to consider placing a UTM firewall on the network pathway between the corporate WAN and the SCADA facility LAN, as extra protection against threats coming across the corporate WAN (which, as we have discussed, is a serious attack pathway). But we will discuss other ways of isolating your SCADA system later in this chapter. If your SCADA system has a WAN connection to another system, and that connection goes over the Internet, then in addition to using VPN technology to secure that connection, it would be reasonable to place a UTM firewall in that connection (after the VPN gateway) to block cross-Internet attacks.

Proxy Server Firewalls (a.k.a., application-level proxies) provide isolation between an interacting client and server. The proxy firewall accepts a TCP session connection with the requesting client and then establishes a separate TCP session connection with the server. The client and server never actually directly communicate. The proxy server is protocol-aware of the application-layer protocol being used (e.g., ftp, http) and performs both normal stateful TCP packet-filtering functions, as well as inspection of the message traffic at the application level, to ensure that it conforms to the IETF specification (a.k.a., stateful protocol analysis or SPA). The proxy receives, inspects, and then regenerates IP datagrams travelling in both directions (fig. 11–2). Proxy servers do not check the data/payload section of the datagrams, just that the application protocol rules are being observed (so they won't discover and block malware). The major problem today, in using an application proxy firewall in a SCADA application, is that proxy server capabilities are only available for well-known IT protocols. To date, there are no proxy server firewalls that handle industrial IP protocols. A proxy server firewall is most commonly applied between Internet-based clients and corporate public servers (e.g., Web sites and email exchange servers). It is unlikely that a proxy server firewall would be applicable to a SCADA system.

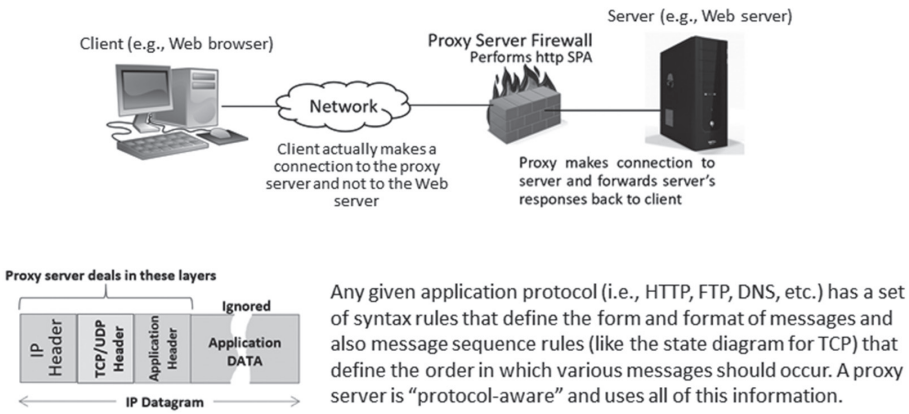


Fig. 11–2. Application proxy firewalls

Deep packet-inspection firewalls go a step further than merely checking protocol rules. A deep packet-inspection (DPI) firewall is like a proxy server on steroids. It doesn't just examine each datagram or fragment for application protocol violations, it reassembles the entire datagram and scans it for malicious content. Deep packet inspection is often used to ensure that data is in the correct format and to check for malicious code/malware. There are multiple headers for IP packets; network equipment only needs to use the first of these (the IP header) for normal operation, but use of the second header (such as TCP or UDP) and port number information allows recognition of the protocol being used. DPI is similar to network intrusion detection, but is rarely used other than at the enterprise level, due to the cost and because DPI can create latency issues (delivery delays), due to the fragment collection and datagram reassembly (and then fragmentation and retransmission). And, of course, malware signatures need to be maintained, same as with a UTM firewall appliance.

Stateful packet-filtering firewalls are the most basic form of firewall and, due to the available CPU processing and memory capacity of smart devices today, this functionality has been embedded in devices that are not traditionally considered as a firewall, such as Ethernet switches and routers. With this type of firewall, you establish a set of "rules" concerning IP datagrams that are allowed to pass through the firewall (one set for incoming [ingress] and a separate set for outgoing [egress]), and as each IP datagram arrives on an interface, the rules are applied and the datagram is either passed or deleted. Packet filters mainly look at the header information in the IP header and TCP/UDP/ICMP headers and possibly also the Ethernet frame header. Also, they make pass/discard (permit/block) decisions on an individual packet basis with no concern for prior packets (fig. 11–3).

Having said that, with stateful inspection (SI) added to basic packet filtering, the firewall does keep track of the current TCP state of every active session/connection, and if either of the communicating parties violate the TCP state-machine rules, the session will be shut down (fig. 11–4). Although UDP is stateless, a firewall doing SI usually makes a note of outgoing UDP messages, with the expectation of a possible returned one (from the recorded destination IP address) within a reasonable time frame. Many of the early malicious methods for crashing a computer's IP stack and networking software were eliminated with the introduction of SI firewalls. In an earlier chapter, we mentioned using firewalls to segment various portions of your SCADA system LANs and WANs and to control and restrict message traffic. Packet-filter firewalls with SI (and well-designed rule sets) are generally an effective (and cost-effective) choice for such network/LAN segmentation, especially if combined with a traffic monitoring technology such as NIDS.

Many of today's Ethernet switch products, because of excess processing capacity and extensive memory capacity, have been extended to add features to enhance their functionality and cost-effectiveness. One capability popping up in all but the lowest-capability switches is the ability to have packet-filter rule sets that can be used to control and manage traffic on a port-by-port basis. A major manufacturer of router,

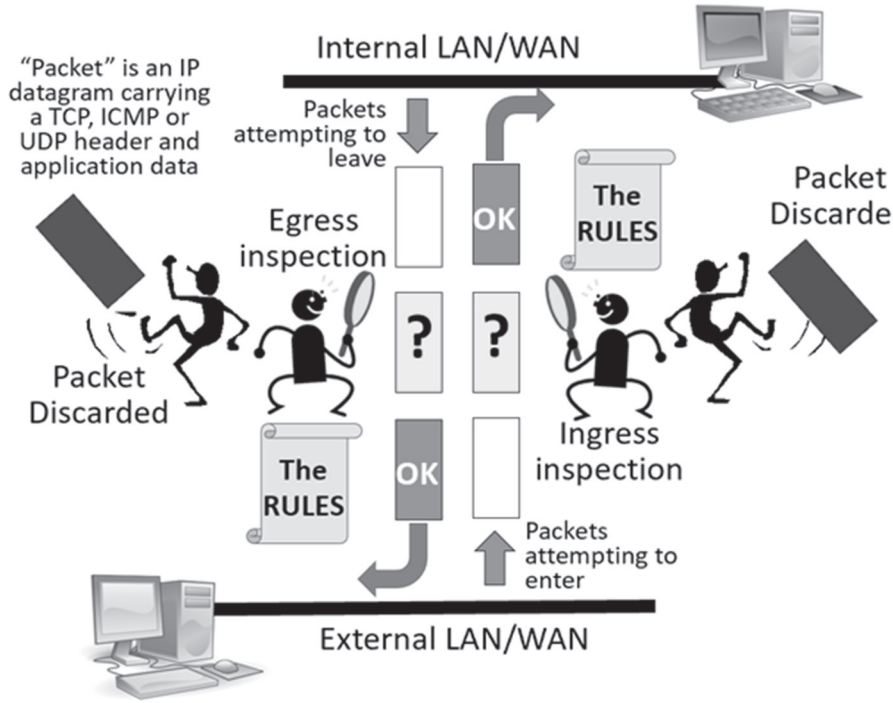


Fig. 11–3. Packet-inspection firewall

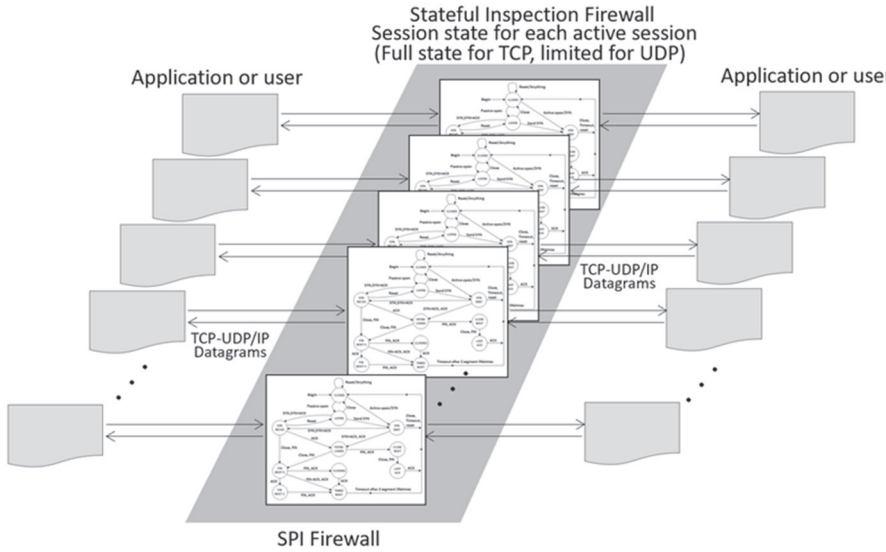


Fig. 11–4. Tracking TCP state for each session

switch, and firewall products allows you to define a set of rules for arriving Ethernet frames (carrying IP datagrams) that will either permit them to pass through the switch or cause them to be discarded (fig. 11–5). The switch supports typical "permit" and "deny" rules for a wide range of IT protocols and even has extensions for filtering based on times and dates. (The terms *access control list* [ACL] and *rule base* are also used in reference to the rules configured in a firewall.) When we discussed the ISO/OSI seven-layer model, we mentioned that Ethernet switches normally only deal with MAC addresses and thus are layer 2 (data link-layer) devices. But a switch that supports packet filtering deals in IP addresses and port numbers and so would be more of a layer 3 (network layer) device. You may hear such a switch called a layer 2/3 device because of this. Since in most instances, each device/computer gets its own private connection to an Ethernet switch, this particular type of switch lets you have rules that are device/computer-specific and could eliminate the need for a separate firewall on the network. In effect, this type of switch lets you segment your LAN on a per-device/computer basis, specifically defining the computers and devices with which it is permitted to communicate. We have previously mentioned that some firewalls track MAC-IP address pairs (in DHCP message traffic) and use them to detect and block ARP cache poisoning attacks. This requires them to deal with IP addresses, and so again they would be considered as level 2/3 devices.

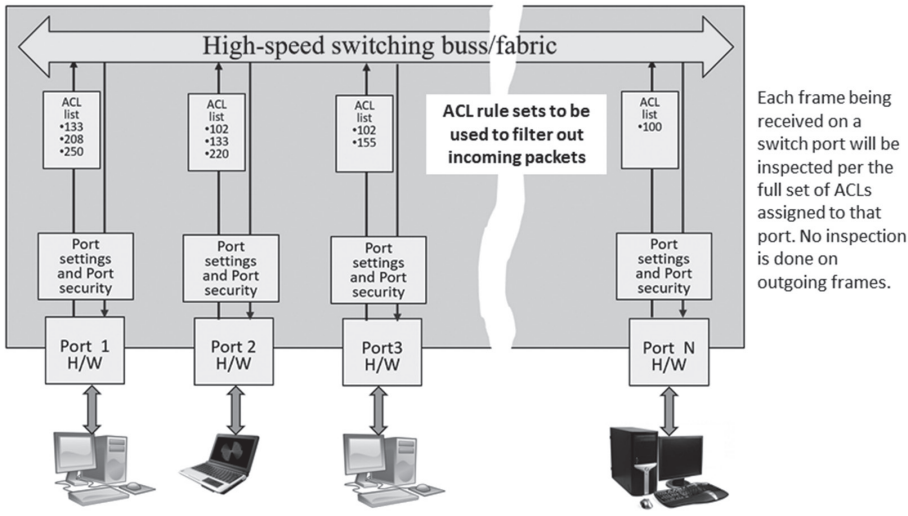


Fig. 11–5. Ethernet switch with packet filtering

Firewalls (even those integrated into a switch) can be very effective if properly configured with the right set of rules. On the other hand, with poorly designed or badly mis-ordered rules, a firewall can be next to useless. With most firewall rule-set definition tools, rules are built and appended to a list *in the order in which you define them*. When comparing a received packet to the rules, they (the rules)

are *checked in that same order*. To give the best throughput performance, firewalls *stop checking any subsequent rules* as soon as they find *the first rule* that applies to the data in the packet. One of the major reasons that firewalls fail is that rules are incorrectly ordered so a rule denying some messages is placed AFTER a rule that permits some, or all, of that same message traffic. For example, with the following rules, Tim Shaw, from Detroit, will be permitted entry, even though there is clearly a rule (or two) that should specifically preclude his entry:

1. Permit anyone from New York	(nope, not from New York)
2. Permit anyone from California	(nope, not from California)
3. Permit anyone with lastname Shaw	**(yes! lastname is Shaw, enter. . .)**
4. Deny anyone fullnamed Tim Shaw	—never even checked—
5. Deny anyone from Detroit	—never even checked—

Rules in the wrong order, or that conflict, are a common reason for firewall failure. Another common reason is rules that are too broad in scope (e.g., there are five computers that ought to be allowed to connect to this computer on port 110, but the one defined rule says "ANY" for the source IP address in order to keep from having to write five separate rules). One final issue is the lack of a final rule to block anything that wasn't permitted by a proceeding rule in the list (e.g., the final rule in the list should always be "DENY ALL").

SANS Institute's Firewall Checklist, under Security Elements, recommends the following order for firewall rules to be applied:

1. Anti-spoofing filters (blocked private addresses, internal addresses appearing from the outside)
2. User permit rules (e.g., allow HTTP to public Web server or SMTP to an Email server)
3. Management permit rules (e.g., SNMP traps to network management server)
4. Noise drops (e.g., discard OSPF and HSRP chatter)
5. Deny and alert (alert systems administrator about traffic that is suspicious)
6. Deny and log (deny and log all remaining traffic for analysis)

Most (but not all) firewalls also have either a *default permit* or *default deny* setting, so that if none of the defined rules are found to apply, the packet will be either permitted or dropped, based on that default setting (dropped/deny is the best choice). Some of the more capable firewalls, upon detecting malicious traffic, can dynamically create a new rule that blocks the source of the malicious traffic (datagrams coming from the sender's IP address). This is often referred to as a *dynamic policy* functionality. Firewalls vary a bit from manufacturer to manufacturer, but most allow each rule you define to specify the following sorts of information:

- Name: Name given to the rule. This name should describe the policy the firewall rule is being used to implement (e.g., Allow Web browsing, Block file transfers).
- Interface: Select the interface that the firewall rule will be applied to (e.g., Any, Dialout/Cellular, VPN, Network Interface, FastEthernet1/1).
- Port Range: Specify the port or range of ports (e.g., 1000–1500) that the rule will apply to. "Any" means all ports.
- Source MAC address: Specify the source MAC address to be matched. MAC addresses use the format XX:XX:XX:XX:XX:XX, where XX are hex digits.
- Source Address Range: Specify the source IP address (or address range) to match. IP address ranges typically use the format - ip/netmask. "Any" means all IP addresses. Unless this is for a Web, email, or DNS server, then it really should not be set to "Any" in most cases.
- Destination Range: Specify the destination IP address/address range to match. IP address ranges typically use the format - ip/netmask. This really should not be set to "Any" in most cases.
- Protocol: Select if the firewall rule will apply to "TCP" or "UDP" or "TCP and UDP" or "ICMP or ESP or GRE" (a combination of multiple protocols) or use "Any" for all protocols
- Direction: Select the traffic direction that the firewall rule will apply to (Ingress = incoming or Egress = outgoing).
- Action: Select the action (Permit/Accept or Deny/Block) that will be applied to the packets detected that match the Interface+ Port Range+ Source/destination Address Range+ Protocol+ Direction.

The following is an example of a rule definition for a Cisco firewall device. The rule is being added to group (access-list) 121 and is a rule that permits an incoming TCP connection from any source IP address to the destination computer with IP address = 192.168.27.3 into the port assigned to smtp protocol (25) on that computer; in other words, for incoming email from other (unknown in advance) email exchange agents:

access-list 121 permit tcp any host 192.168.27.3 eq smtp

Depending on the vendor, you may be able to configure the firewall using a graphical configuration tool, or you might have to define them interactively using the firewall's console port or command-line-interpreter. There are third-party tools that can build rule sets for various firewall manufacturers products. Some vendors allow you to create rule groups and then assign one or more of these rule groups to particular interfaces. Firewalls, especially the more sophisticated ones, such as an Enterprise or Next-Gen firewall, have a number of configuration options, ranging from setting up VPN and NAT functions to enabling (or disabling) remote administrative access. Cybersecurity and convenience are often at odds with each other when making

configuration choices, particularly remote administration choices. Being able to sit at your desk and administer all your firewalls is very convenient, but it also opens the possibility of an attacker doing the same thing. Unlike an IT environment where firewalls may need more frequent attention, in a SCADA system, it is unlikely that firewall rules will need updating very often, especially if your rules permit specific traffic and block everything else, because new unexpected, unforseen stuff will probably be blocked. You can make your firewalls less vulnerable to being compromised if you make them "transparent." This means not assigning an IP address and subnet mask to any of the interfaces and using the console port when any administrative function is necessary (fig. 11–6). Without an IP address, no one can send (malicious) messages to (a.k.a., attack) the firewall, only to devices/computers on the other side of the firewall, which one would hope the firewall will block. Incidentally, the same thing applies to your Ethernet switches. Without an IP address, they also cannot be remotely attacked. A transparent firewall should not typically sit between different sub-domains (groups of computers with a different network portion of their IP address), since the firewall can't provide inter-domain routing. If a firewall needs to perform routing, then it needs an IP address on its interfaces, and the firewall's IP will probably be specified as the default gateway for computers on each sub-domain.

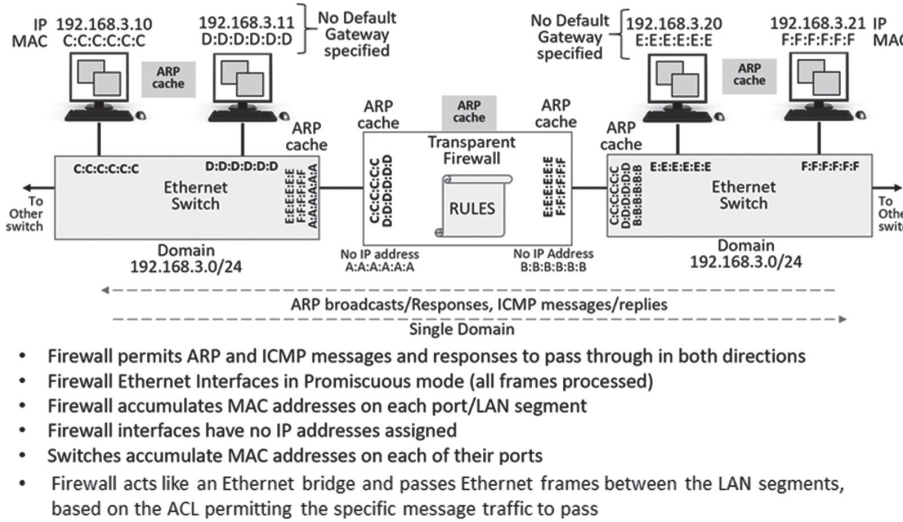


Fig. 11–6. Transparent firewall operation

In transparent mode, a firewall is literally invisible to the devices on the various network interfaces. It looks more like an Ethernet bridge than a conventional routing firewall. (A routing firewall provides message delivery between the LAN segments connected to its interfaces [i.e., routing services] and is usually designated as the default gateway for routing to IP addresses not local to the connected LAN

sub-domain), Transparent firewalls still pass ARP and ICMP messages between the LAN segments, but they never treat any packets/frames as being directed to themself, so you can't administratively interact with the firewall or send it attack exploits. This is particularly important if you are running firewall software on a Windows or Linux OS platform as those operating systems may have exploitable vulnerabilities. Packets must still match a "Permit/Allow" rule to pass through the transparent firewall. A firewall that has IP addresses on its ports can provide inter-domain routing, and, in fact, each port (and a firewall can have many, not just two, even if that is how they are commonly drawn) can be connected to separate sub-domains.

Host-based or "personal" firewalls are useful for protecting individual computers from cross-network attacks. Windows has come with a built-in personal firewall (the Internet Connection Firewall) beginning with XP and Server 2003 (then renamed Windows Defender™ as of Windows 10). Of course, there are many third-party firewall applications available for Windows as well. Linux has included "iptables" since forever, but more recently, most distributions include "firewalld," which is less complicated to configure. (Ubuntu comes with "ufw"—the uncomplicated firewall.) Host-based firewalls have access to the computer resources and operating system of the computer on which they run, which is not possible for an external firewall. Because of this, a host-based (personal) firewall can take actions unavailable to a conventional, external firewall, such as controlling which programs can and cannot access the local network and/or Internet and providing the user with information about any application that makes a connection attempt. In other words, a host-based firewall can function somewhat like a HIDS or endpoint protection application.

Today, with IP networking going out to the field sites of a SCADA system, there will be field equipment (e.g., PLCs, RTUs, etc.) communicating with the SCADA system using IP-based *industrial protocols*. It is possible to place a firewall at each such field site to protect against attacks coming across the network (and up at the SCADA system connection to that WAN, as well), but conventional firewalls do not understand industrial protocols and would only support rules that protect based on IP address and port numbers, both of which can easily be spoofed. A new category of firewall is now available that can be made *protocol-aware* of several popular IP-based industrial protocols (fig. 11–7). By this, we mean that these firewalls can have rules that go down to the protocol function code and data object level. Using Modbus/IP as an example, the firewall rules can, in addition to permitting/blocking based on IP address and port numbers, permit specific Modbus function codes (e.g., Read_Input_Registers : function code 4) and register addresses and block everything else. These types of firewalls are being used in many industrial automation applications and support IP-based protocols such as OPC (classic), Modbus/IP, EtherNet/IP (CIP), BACnet, GOOSE, and DNP3—and more protocols continue to be added each year. One of the earliest manufacturers of industrial protocol-aware firewalls is Tofino Security (https://www.tofinosecurity.com), a Belden corporate division. Their firewall is fully transparent; there is no option for giving it an IP address on either of its two interfaces. They use

a simple means of communicating with their firewall: you send messages through it, addressed (the datagram's destination IP address) to a computer or smart device on the "protected" side, with a destination port number set to a particular value, and the firewall recognizes that as a message to itself due to the destination port number and processes the message without passing it onwards. Of course, in theory this scheme provides a mechanism for attacking the firewall (just set the right destination port number in your messages), but the messages exchanged between the Tofino configuration tool and the Tofino firewall are encrypted. You can also configure these firewalls using a USB flash drive if you are concerned about cross-network communications. That firewall has a unique feature—a test mode in which it blocks no message traffic at all but it sends Syslog messages to a designated server indicating messages it WOULD HAVE blocked, were it in normal operating mode. This is a very useful feature for testing out the firewall without actually accidentally blocking important message traffic.

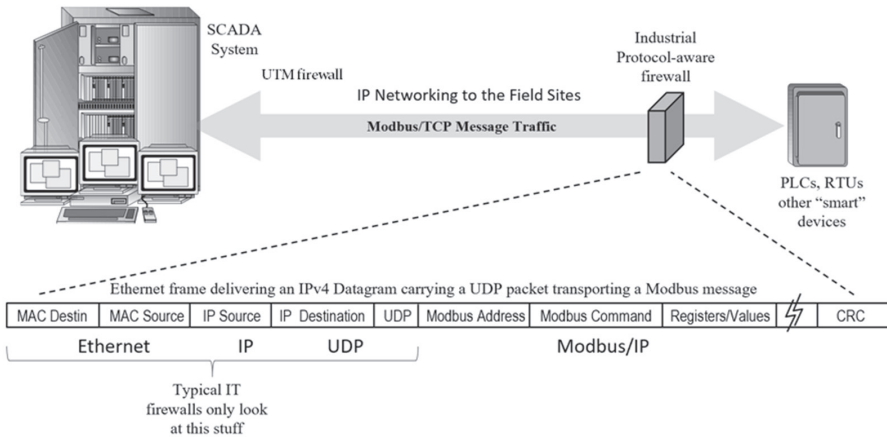


Fig. 11–7. Industrial protocol-aware firewall

The industrial protocol-aware firewalls basically place a logical barrier between the field device and the SCADA system (or an attacker), which blocks access to data and control objects configured in the field devices and limits the functions/commands that can be permitted to be delivered to the device. The firewalls do not place restrictions on outgoing messages (the field device can send applicable responses to any received commands), but incoming messages are strictly filtered. I like to picture this as a metal shield placed in front of the field device with cut-outs for the specific data elements permitted to be accessed and shaped to allow only specific commands through (fig. 11–8). Note that this doesn't prevent an attacker from sending commands to those same data objects (Modbus registers in the example shown), as long as they spoof the permitted sending IP address and figure out which command (Modbus function code, in the example shown) is permitted. Of course, the response from the field device will be directed to the SCADA system's IP address and will not be received by the attacker,

so the attacker won't know if the command actually worked. Attempting to try various commands and register addresses (in the case of Modbus/IP) to figure out the firewall restrictions will result in a flood of Syslog messages being sent to the SCADA system (or SIEM?) from the firewall, thus alerting you of the malicious activity. (Think of it as any attack that hits the shield, rather than a cut-out, will make a noticeable clang!) And, of course, if port security is enabled and a NIDS is monitoring network traffic, it is likely that any attempt to connect into the WAN will be noticed and alarmed.

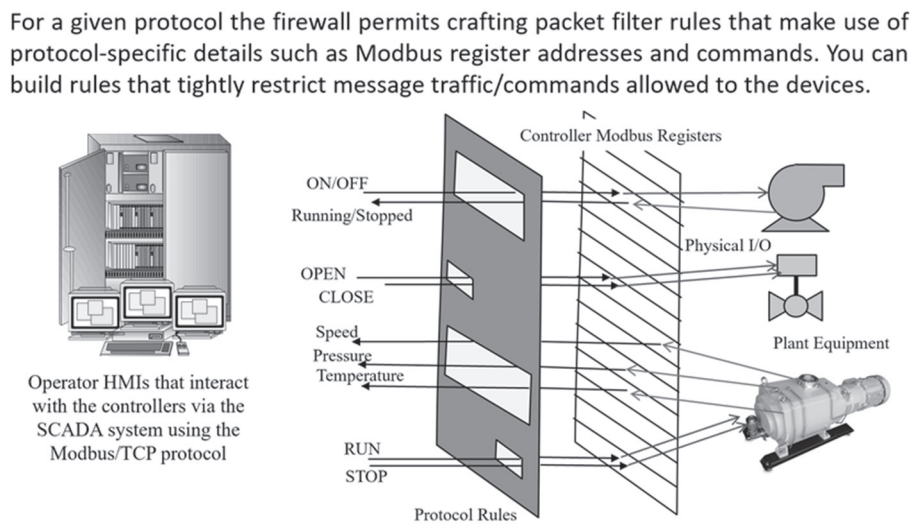


Fig. 11–8. Industrial protocol detailed filtering

Although an industrial protocol-aware stateful packet-filter firewall might be the right solution for protecting field devices connected to the SCADA system via IP-networking, it is not the right solution for defending the SCADA "end" of that communication pathway. That is because an adversary who gains network access is going to attack the SCADA computers with conventional cyberattacks, exploits, and malware that are better identified and blocked using either UTM (or possibly Next-Gen) firewall technologies. Also, because the SCADA system will be communicating with a large number of field devices, possibly using several industrial protocols and conventional IT protocols, a more conventional firewall would be a better choice. The current industrial firewalls are not intended to support multiple concurrent communication streams, and although capable of stateful packet inspection, they do not have the more advanced features of a UTM or Next-Gen firewall.

One of the challenges with SCADA systems is that, by design, they are usually applied to monitor and remotely control processes that are geographically distributed, potentially over a very large area, and it is not always easy to provide good physical security at every field site. With an IP-based WAN architecture, that

means that there is a risk that an adversary might find a location where they could gain physical access to network communication equipment and, at that point, have network communications connectivity to other field sites, as well as to the SCADA master/host. Remember that the great flexibility and power of IP-based networks is that they are designed to allow connectivity from any/every point to every other point. Another issue is that commonly used IP-based industrial protocols are well known and documented, and software for such communications is readily available. This means that an adversary could easily spoof message traffic to field devices and the SCADA host computers (although an industrial protocol-aware firewall at the field sites can protect against such spoofing, as noted above). It has already been noted that corporate networks are notorious for being penetrated. If there is no adequate isolation between your corporate network and the SCADA system's network, then an attacker can use that connectivity as an attack pathway into the SCADA system. We will shortly discuss ways to block that attack pathway.

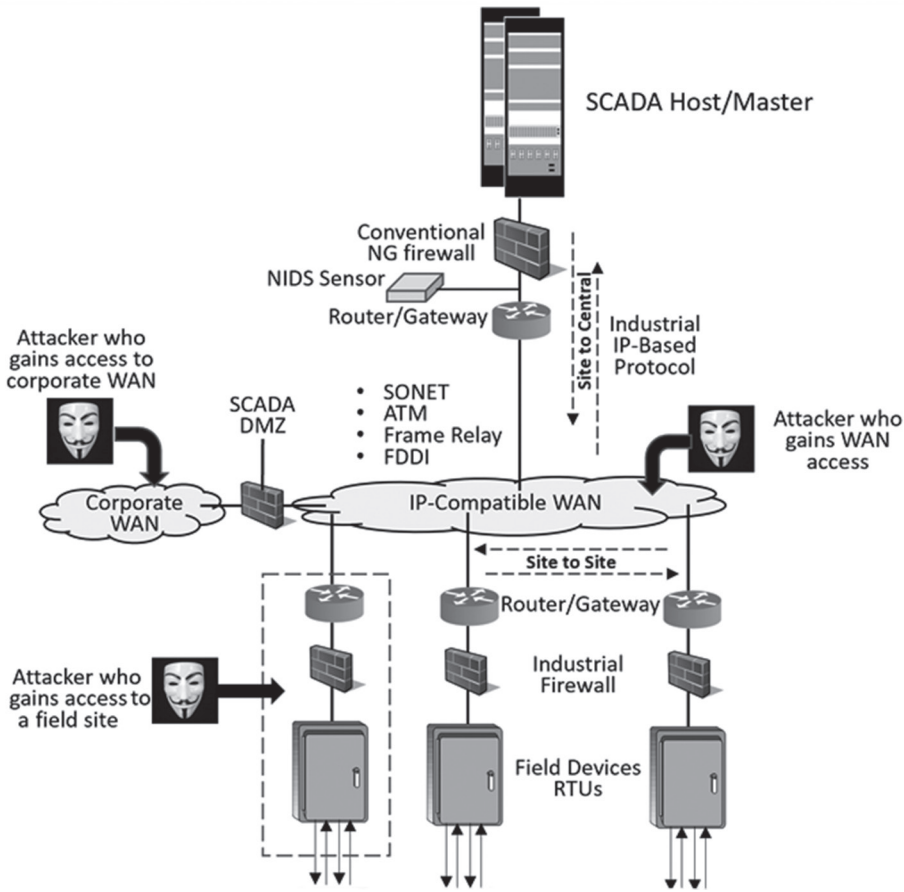


Fig. 11–9. Cross-network vulnerabilities with IP to the field

Network intrusion detection (and prevention) systems

If your physical security and protections are adequate, an attacker is not going to be able to physically get to your SCADA system and thus will be forced to attempt remote, cross-network attacks or to perpetrate some social engineering ploy. As has been mentioned, it is highly likely that an attacker could gain a foothold in your corporate WAN, and, most likely, that provides a potential attack pathway to your SCADA system. One way of monitoring (and protecting) that pathway (and any IP-network connection to the field sites), in addition to using a firewall, is to employ a network intrusion detection system, or NIDS. Depending on how you deploy this technology, it can also be capable of shutting down any detected attack and not just detecting (which makes it an intrusion prevention system or NIPS). A NIDS typically consists of one or more *sensors* and a management console for configuring the sensors and where alerts from the various sensors can be displayed and correlated (fig. 11–10). For a NIDS to perform properly, it needs to see all of the traffic from around the network you are trying to protect—and especially on network connections that could be attack pathways (e.g., to the corporate WAN, to the field sites, to regulators/vendors/customers whose systems you don't control). Depending on the product, the sensors may perform signature- or/and anomaly-based detection, and they may pass suspicious messages to the management console for more advanced analysis. Not all NIDS require a management console once the initial configuration and training of the sensors has been completed; other products require the console as part of the overall detection mechanism, particularly where multiple sensors are used to monitor different portions of the overall network traffic.

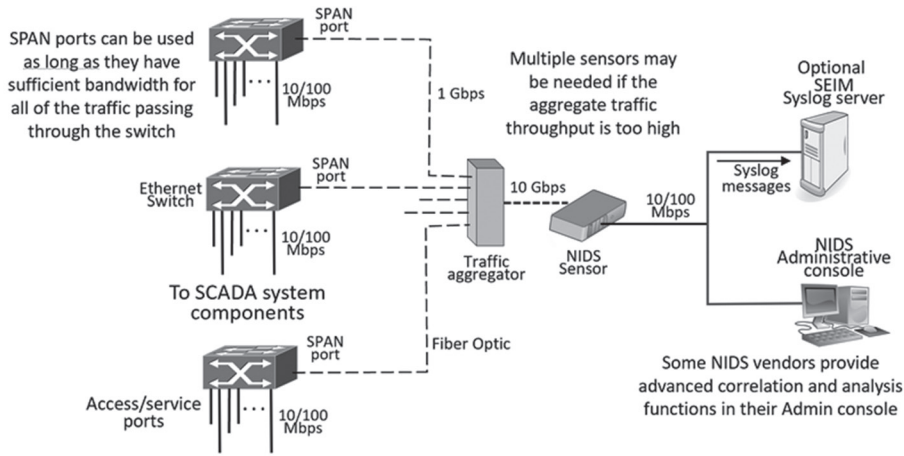


Fig. 11–10. NIDS components and structure

One way to collect such traffic is to use the SPAN (mirror) ports on your network switches to copy all Ethernet frames passing through that switch and deliver them to one or more NIDS sensors, based on the aggregate traffic bandwidth. A

SCADA system is not an IT server farm, with the kind of message traffic that hundreds of active users, pounding away on keyboards, can generate, and so it may be that a single NIDS sensor could be adequate. In which case, the SPAN ports from the various LAN switches can go to an *aggregator*, which will then combine and deliver all the frames to the single sensor. (An aggregator is actually just another switch). A NIDS can fail to properly detect attacks if you don't design the system with adequate bandwidth (from your switches to the aggregator) and processing power (in the aggregator and sensors), because messages may be dropped and incomplete traffic might prevent detection. A switch will just discard frames if it fills its buffer memory, and this can degrade the ability of a NIDS to perform its detection function since it is not seeing everything. (And if message frames are arriving faster than the sensor can process them, the NIDS ability to detect will similarly be degraded.) Thus, SPAN ports work best when they have a higher bandwidth than the access/service ports used to connect with end devices (e.g., if the SPAN port is 1 Gbps and the access ports are all 10/100 Mbps). Note that the ability of a switch to support mirroring/SPAN differs from vendor to vendor and sometimes from product to product. Some switches permit only one port on the switch to be copied to another port as a mirror. This is particularly true for switches where all the ports support the same speed (e.g., a switch with eight ports and all of them are 10/100 Mbps). Also, NIDS sensors usually have a frames/sec rating on their processing capabilities, and they perform best if you never go above 75 percent of that rated capacity. In setting up SPAN ports it is important that you don't create a situation where the same Ethernet frame is replicated multiple times to the NIDS as this can appear as malicious activity. Only the access ports of a switch should be included in the SPAN (never trunk ports) and then only for either incoming or outgoing frames (not both). This will prevent frame duplication.

Another way to collect and aggregate network traffic is by using *network taps* (fig. 11–11). A tap is a passive device that is placed on an Ethernet connection between two devices (or groups of devices, or a server and a group of clients), and it copies the Ethernet frames going in both directions on that connection out to the sensor (or to an aggregator collecting Ethernet frames from multiple taps). Taps are one-way devices; data flows out of the tap, but it is impossible to send frames back into the tap since none of the transmit circuitry is wired. Taps generally support whatever bandwidth is needed, up to multi-Gigabit data rates, and don't have bandwidth limits since they don't perform any message processing. Taps can be purchased with conventional 10/100 RJ45 interfaces, as well as with single-mode fiber optic interfaces for the highest possible bandwidth. The main issue with taps is finding the ideal network connection points in which to insert taps such that all traffic is being recorded. Another issue is making sure you place them such that multiple copies of the same Ethernet frames don't get collected, as that can potentially confuse the NIDS. This can happen if you accidentally tap along a communications pathway in multiple locations.

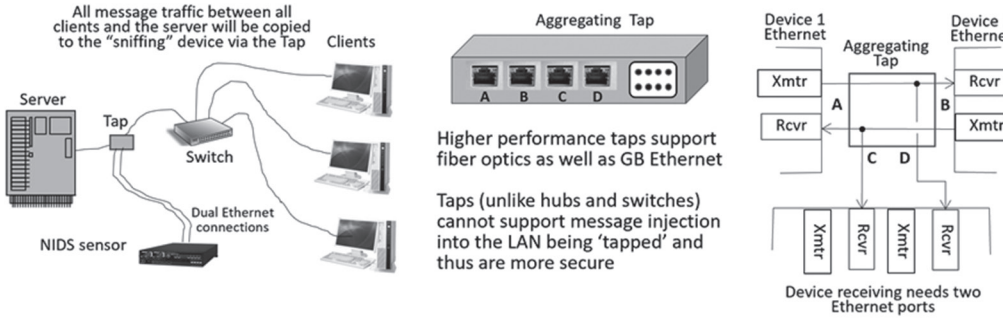


Fig. 11–11. Using network taps for message collection

Regardless of whether you use SPAN ports and aggregators or Taps and aggregators, neither of those technologies allows the NIDS sensor to send messages back into the network being monitored. So, you need to run a separate LAN connection to the sensor(s), which connects them to the management console (and possibly the SIEM, unless the management console will be forwarding messages to the SIEM). Some NIDS products depend on having an active management console at all times (and having that console feeding the SIEM); others only require the management console for initial configuration of the sensors (in which case, the management console function can just be application software loaded onto a convenient computer), and the sensors directly feed messages (as Syslog transmissions) to the SIEM.

The NIDS sensors are similar to a firewall appliance in that they are a high-performance, stand-alone computer that will be configured with rules and signatures for identifying malicious traffic and malware propagation (which also means that they will need to be updated on a routine basis, to keep signatures up to date). Most NIDS sensor products can also perform real-time SSL decryption if needed. Some NIDS products can also watch for anomalous message traffic—traffic that is not obviously malicious, but which is outside the range of normally observed message traffic (e.g., a new protocol never seen before, new source IP addresses never seen before, messages between devices that have never communicated previously). This is a form of anomaly-based detection, and it generally requires that the NIDS monitor and accumulate statistical information during a *learning/training period* so that it has a baseline of "normal/usual" traffic against which to compare. *Anomaly detection* tends to generate more false positives (alerts when there is actually nothing amiss) than does signature-based detection, but it can potentially detect malicious activity for which no signature or recognizable pattern exists. Some NIDS products now include rule sets for various industrial protocols as well as for conventional IT protocols. The vendor-provided rules should also support a wide range of known exploits and attack techniques. When a sensor detects potentially malicious traffic, it will send an alert to the management console, and possibly also a Syslog message to a SIEM. But, if the sensor is capable of inline operation,

then it can do more than just send an alert (fig. 11–12). In that case, it can take an action to block the attack and thus can be described as an intrusion detection and prevention system. The typical action is to *blacklist* the source IP address, drop all datagrams/packets/frames coming from that source, and terminate the TCP session, if there is one established. And of course, as with basic detection-only NIDS, it will send the alerts and Syslog messages as well. When configured as an inline device, it becomes even more essential not to exceed the processing capacity of the sensor, and, of course, the sensor can only watch the traffic on the network pathway in which it is positioned. But with inline positioning, the sensor can usually send messages to the SIEM and management console with no need for a separate LAN connection.

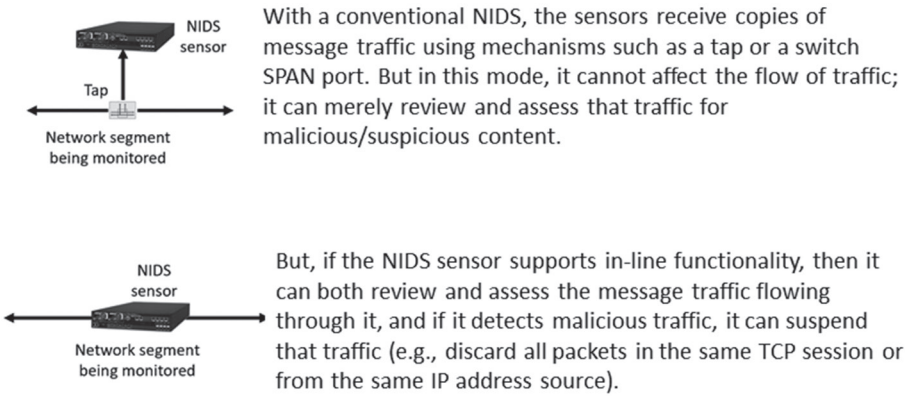


Fig. 11–12. Network intrusion detection and prevention system

Unlike when connected using a tap or SPAN port, if the sensor fails when used inline, the pathway it was monitoring will be disabled, but this is no more likely than having a firewall or switch fail.

There are many commercial NIDS/NIPS products on the market, but for those with some computer skills, a very popular NIDS/NIPS package that is open source, widely supported, and freely available is something called Snort™. Snort was developed by Sourcefire (now a Cisco subsidiary), and it is a free and open source network intrusion detection (NIDS) and prevention system (NIPS)—although there is a commercially supported version as well (https://www.snort.org). Snort has the ability to perform real-time traffic analysis and packet blocking/dropping on IP networks. Snort performs protocol analysis, content searching, and content matching, based on signature definitions. Snort can be used to detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, common gateway interface (CGI), buffer overflows, server message block probes (SMB), and stealthy port scans. Snort can be configured in three main modes: sniffer, packet logger, and network intrusion detection. If configured "inline," Snort

can also act as a NIPS. Snort is comprised of two major components: a detection engine that utilizes a modular plug-in architecture (the "Snort Engine") and a flexible rule language to describe traffic to be collected (the "Snort Rules"). The Snort Engine is distributed both as source code and binaries for popular Linux distributions and for Windows. Third-party tools can be integrated with Snort to increase its functionality and compatibility with other tools and products. You basically take a computer with a Windows or Linux operating system, harden it, and then load and run Snort. The DHS funded a project to develop Snort rule sets, several years ago, for several industrial protocols (e.g., Modbus/IP, IEC-60870-5-104, and DNP3), and these, along with a huge public library of other rule sets, are available to download. Snort has a sophisticated rule definition language that is regularly updated with new features. Rules can be activated by other rules and can be strung together to deal with complex message sequences. Rules can check and compare message content (signature identification) as well as the values of any of the message fields. Snort supports a long list of useful rule options, which can be combined to create very powerful rules. Snort can also forward Syslog messages to a SIEM as part of its detection activities. However, SNORT is a do-it-yourself kit version of NIDS (unless you go with the commercial version), and though it is very powerful and flexible, has a large installed base and good technical support, it may not be right for your organization, depending on the capabilities (and availability) of your IT department and system programmers.

Host intrusion detection (and prevention) HIDS

A NIDS watches for, and potentially protects against, cross-network attacks. But it won't help to defend a computer that has been infected with malware from portable devices or computer-readable portable storage media, whether that happened accidentally or intentionally. And it won't protect against or detect someone attempting the local exploitation of a computer. However, there are technologies for both detecting a compromise and even for blocking a compromise of a computer. HIDS technology comes in several variations, based on how the technology detects system changes and malicious activity. A HIDS monitors the computer on which it is installed, (possibly) analyzing traffic (in and out of the computer), looking for unexpected changes, and logging malicious and suspicious behavior. A HIDS gives you visibility into what's happening on your critical servers, PCs, and workstations (fig. 11–13). With it, you can detect and respond to malicious or anomalous activities that are discovered in your environment. But it can't see what's happening on other computers or on your networks. This is why a HIDS is typically deployed by putting agents on every computer and then linking them to an administrative console that will collect and correlate data from the various agents and why the best security comes from a combination of both HIDS and NIDS technologies. One particular type of HIDS, called 'whitelisting', can both detect and block malware infections and the execution of malware. Many cybersecurity solution vendors offer

both types of technology. Some HIDS products incorporate multiple detection techniques, while others focus on a few or just one. For example, several products only analyze the computer's network traffic, and other products only check the integrity of a computer's critical files and data objects. Specific techniques commonly used in various HIDS products include the following:

- Code behavior analysis: Monitoring programs for behavior that is different from "normal" (anomaly detection) or that is indicative of malicious activities, such as accessing protected system functions, driver installation, improper memory/stack access, or loading DLLs that provide root-level functionality.
- Whitelisting: Pre-specifying allowed executable code (including scripts) and using operating system *hooking* to catch and block any attempt to execute code not previously whitelisted. (Windows has a function called AppLocker, which is a form of whitelisting, but it has some limitations and known bypasses.) Some of these HIDS/HIPS products also provide some form of application memory protection to block buffer overflow style attacks.
- Network traffic analysis: Monitoring for unusual/restricted network connection requests, scanning incoming messages for malware signatures, monitoring email and Web browser activities. If used as an IPS, then it may restrict access to certain Web sites and message traffic from specific IP addresses, as well as service/port access.
- File system monitoring: Periodic file integrity checking (comparison to pre-computed file hash values stored in an encrypted cache), file attribute checking (change of owner/group, R/W/X attributes, permissions, timestamp), file access attempts. It may also monitor for registry changes and modification of critical system files.
- Log analysis: Analyze O.S. and application logs to identify malicious activity, such as shutting down or starting an application or service, successful and failed authentication (login) attempts, and security policy changes. Of course if your logs are being forwarded to a SIEM then this same sort of log analysis would be performed by that system and so a different type of HIDS might be preferable.
- Network configuration monitoring: Network interfaces being placed in promiscuous mode, additional TCP or UDP ports being used on the host, or additional network protocols being used, such as non-IP protocols.

The whitelisting version of HIDS is of particular interest as it actually prevents the execution of any form of program code (including scripts and BATCH files) that has not been pre-approved. This actually tends to stop most (but not all) types of cyberattack in their tracks. Other forms of HIDS, such as the file monitoring variations, alert you that you have been compromised, but do not actively block compromise attempts. A combination of both technologies is a very good defensive strategy.

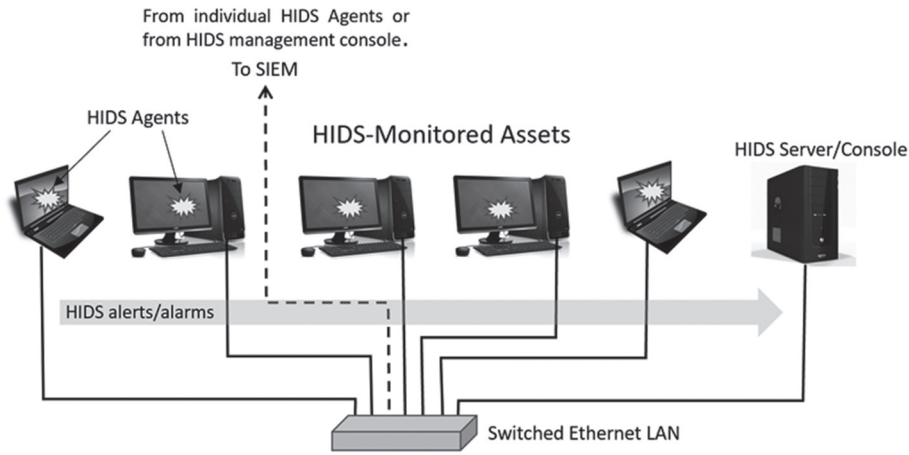


Fig. 11–13. Host-based intrusion detection

The term *endpoint protection* (or *endpoint security*) is often used to describe various HIDS/HIPS products combined with other technologies such as AV scanning and personal firewall capabilities. Endpoint protection capabilities may include performing serial number checking on inserted USB flash drives to prevent use of unauthorized portable media (*USB whitelisting*), and it may include the automatic AV scanning of any file read from any portable media. Some endpoint security solutions also integrate a personal firewall that may support email and Web-browsing protections (which should not be an issue for the SCADA system.) One HIDS product (Tripwire Enterprise©) states that it can automatically review all the files that have been added or modified and both automatically repair (undo) the changes and notify you of the situation. Thus, it can somewhat act as an intrusion prevention (or at least restoration) system. There are a number of companies offering endpoint protection solutions, including the following (in no particular order):

- Webroot (Windows, Mac, Linux, cloud)
- Kandji (Windows, Mac, Linux, cloud)
- Malwarebytes (Windows, Mac, local install)
- AVG (Windows, Mac, Linux, cloud or local install)
- McAfee (Windows, Mac, Linux, cloud or local install)
- Avast (Windows, Mac, Linux, cloud)
- Avast Endpoint (Windows, local install)
- Kaspersky (Windows, Mac, Linux, local install)
- ESET (Windows, local install)
- Symantec (Windows, Mac, Linux, local install)

Clearly the products that run "in the cloud" (access the vendor's Internet-based servers to perform their functions) are not of much use for protecting PCs or servers

that are part of a SCADA system, because your SCADA system should really not have Internet connectivity or access. But the ones that are locally installed will still require some level of periodic updating to keep current with malware signatures. You will need a periodic procedure (at least monthly) for downloading, transferring, and then installing updates on the various SCADA system components that have malware detection functions (e.g., computers with endpoint security or AV, NIDS sensors, firewalls). If your particular NIDS and/or HIDS products use a management workstation, then those will need to receive the periodic updates.

Deployment of HIDS technology does require installing software (agents) on the computers to be monitored, and the agent will have an impact on the computer as it uses computing power, memory, and disk space. If it is an intrusion prevention system as well, then it will need to be properly configured so as not to block critical SCADA functions. For these reasons, some organizations are uncomfortable about installing and applying HIDS and may just install some form of personal (host-based) firewall software on their critical SCADA computers, servers, and PCs. It should also be pointed out that both HIDS and NIDS, especially if they use statistical or anomalous behavior monitoring, will tend to generate false positives, particularly when first enabled. Many products allow for configuration adjustments (a.k.a., *tuning*) to eliminate specific false positives that keep recurring. Expect that there will be some initial period during which your NIDS/HIDS will generate false alarms and that you will need to make setting adjustments, as required, to eventually eliminate these false alarms. False negatives (failure to detect) can occur as well, and these are problematic as you don't generally know they are occurring. This why having multiple, overlapping detection capabilities is important.

Configuration and network asset monitoring

There are less invasive (than HIDS) security solutions (solutions that don't require an agent installation) available to monitor for unauthorized, unexpected changes to system settings, security policies, and configuration data. Generally, all computers and Ethernet-TCP/IP-enabled devices today (including some PLC products) support simple network management protocol (SNMP—preferably version 3) and contain a management information base (MIB) that incorporates all of those data elements (system settings, security policies, and configuration data), plus diagnostic information and a range of other data, based on the type of device. SNMP support is a very useful tool for monitoring the various network-connected assets in a large system or network. SNMP is a standardized protocol that can be used to read and write data values in a computer's (device's) MIB and, if the device supports this capability, to even send unsolicited notifications (called *Traps*) if specifically monitored values exceed threshold limits set by the user or if the state of a device or monitored value changes (e.g., printer is out of paper, PC disk drive is 85 percent full, frame error rate on switch port #2 exceeded 10 percent)

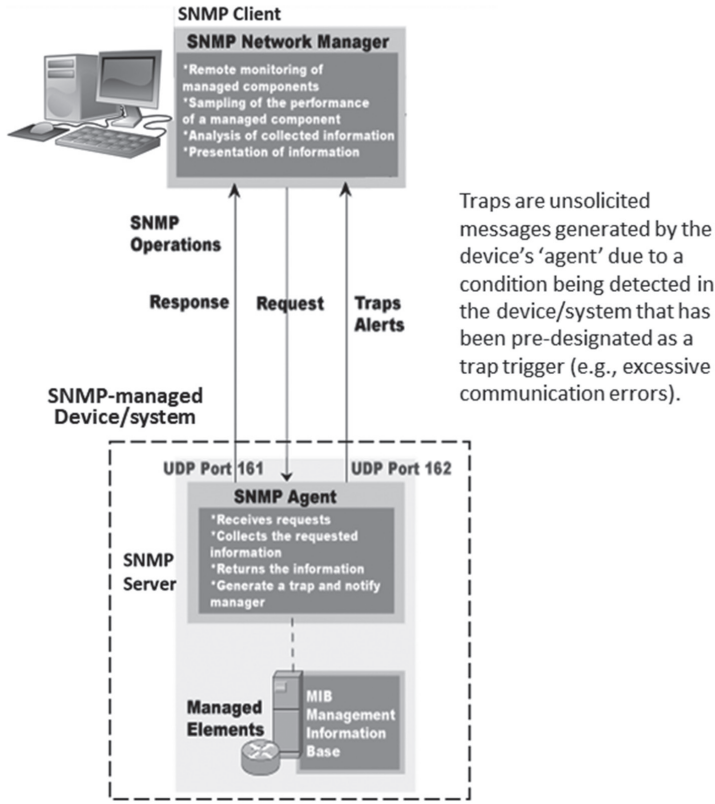


Fig. 11–14. SNMP Client and Agent communications

(fig. 11–14). Although most devices and computers come with SNMP agent support factory-installed (although it may have to be enabled), you have to purchase SNMP client software tools in order to create an SNMP network management console if you wish to use these capabilities. IT organizations regularly use SNMP to monitor all of their network-connected computers, servers, firewalls, switches, routers, printers, and even UPS equipment. An SNMP client package can be used to receive and display traps, but also to periodically fetch operational statistics such as message traffic (and error) counters and display them as a trend. Client software tools can be used to create user-configurable dashboards with a wide range of widgets (plot, bar graph, meter, time-trend, histogram, etc.) that can be used to display individual parameters or sets of parameters showing current and/or historical data. Most such tools are drag-and-drop, so no (or very little) programming is required. Since SNMP is a well-defined standard, these clients work with any devices that supports SNMP. In effect, SNMP can be used to create a SCADA system that monitors your SCADA system. But SNMP has some security issues in versions 1 and 2,

so if you use SNMP, you will want to use v3 due to its use of encryption (because passwords called *community strings* are sent in the read/write messages and, in those earlier versions, the passwords are in clear text and could be discovered), and you will want to make sure to change the factory default passwords (called *community strings*) used for reading and writing, as they default to the words "Public" and "Private" and every hacker knows this. (The SNMP server configuration in each computer and device will need to be given new community strings, and then those will also be configured in the client software.) It is recommended that you don't use the same community string pairs in every device and computer, although that makes managing them a bigger effort. There is no basic reason to change these community strings periodically, as you might your computer passwords, because your SCADA system LAN should be isolated and protected. There are free limited capability (or full capability—limited lifetime) versions of SNMP client software that you can download and try (fig. 11–15), such as from Paessler (https://www.paessler.com) and SolarWinds (https://www.solarwinds.com). As mentioned, most network-connected devices support SNMP, which means that configuration monitoring can be done to your network assets including your firewalls, NIDS sensors, switches, and even your SIEM, which gives you an added level of detection protection. Clearly, there is some effort required to utilize SNMP for SCADA system monitoring, but it is a useful capability if properly implemented. Enabling SNMP support is a potential security risk because if you don't use version 3 and/or don't change the default community strings, then an adversary who gets network access could wreak havoc using SNMP to change settings and configuration data.

For automated configuration change monitoring, there are commercial monitoring products that will upload the entire MIB (issue a **GetBulkRequest** SNMP command, which was added in version 2) of your computers and devices, as a presumed-good baseline, and then periodically re-upload those same MIBs and compare the important element values between the original baseline MIBs and the current MIBs, all this happens with no need to install agent software (e.g., Tripwire CCM™—https://www.tripwire.com). Changes to key settings would indicate an unauthorized modification, and an alert (and Syslog message) would be generated. Very little configuration is required to use these sorts of products, as the vendors have designed them to integrate well with the Windows and Linux operating systems and most popular smart device types (e.g., switches, printers, routers, firewalls). This technique is not invasive (no agent software needs to be installed on the monitored devices, computers, systems), but you do need to enable the SNMP server functions of each. Plus, the periodic SNMP message exchanges will be noted by the NIDS, and so you will probably need a rule to prevent that generating an alarm (or you may allow the NIDS to alert the SIEM and let the SIEM decide if this is expected or anomolous). It could be useful to have the SIEM monitoring for periodic message traffic as a means for verifying that the configuration monitoring function continues to function. At least one major European manufacturer of automation equipment is already integrating SNMP server functionality directly into the Ethernet module of their PLC

products. If you want to examine MIBs and play around with SNMP, there are free *MIB browser* tools available for download. They will allow you to fetch a MIB from a device and examine the various data elements (and even change them). MIB data elements are organized in a tree structure using an assigned number sequence called abstract syntax notation (ASN.1), which is sort of like using number sequences in a document where each subtopic/subparagraph gets an added digit (1, 1.1, 1.2, 1.2.1, 1.2.2, etc.). Fortunately, both SNMP clients and MIB browser software include the textual descriptions for each data object so you don't have to learn ASN.1.

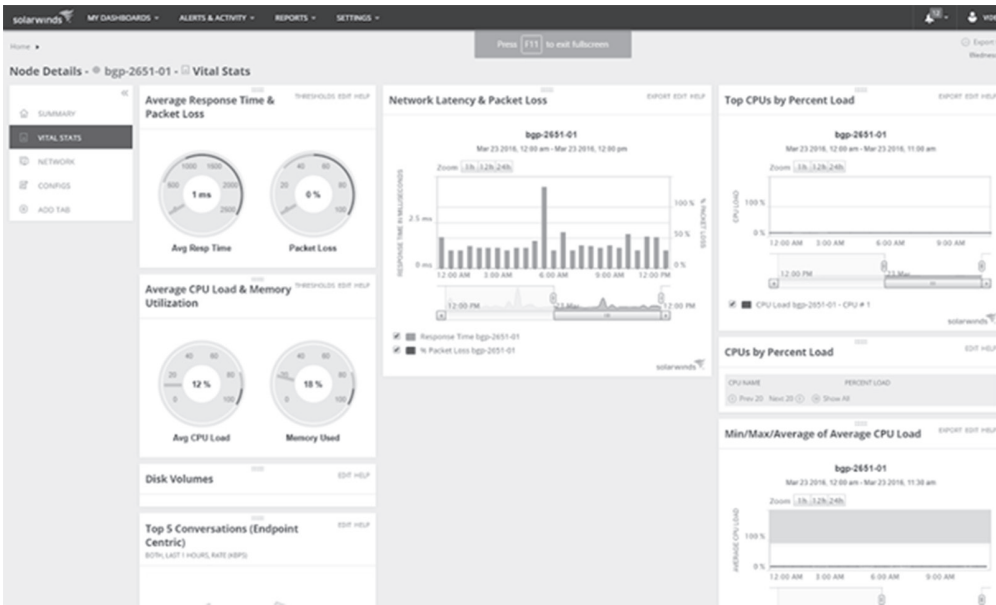


Fig. 11–15. SNMP client software (network monitor) from SolarWinds™

There are also products that perform an automated network scan of your entire LAN/WAN and identify connected assets and then periodically perform this same operation both to identify newly discovered assets and to confirm that previously discovered assets are still functional and communicating. (For example 'WhatsUp Gold' from Ipswitch, Inc. www.whatsupgold.com) These tools use layer 2/3 discovery results (Ethernet and IP discovery mechanisms) to build a detailed interactive map of your entire networked infrastructure. They can monitor and map everything on your LAN/WAN, including devices, wireless controllers, servers, PCs, and even virtual machines. Some products go even further and provide some level of configuration and application monitoring and even traffic flow monitoring (which gives them some functional overlap with the monitoring products described above). Again, these sorts of applications tend to require little configuration effort and provide a useful user dashboard for network status monitoring. But the traffic they generate

will be noticed by the NIDS, and so you will likely need to add rules to the NIDS to ignore the traffic they create (or you can do so at the SIEM level, which might be better because you can have more sophisticated rules). Since network scanning to identify assets is a cyber attack methodology (i.e. 'discovery') you will need to have SIEM rules that differentiate between permitted scanning and unauthorized scanning.

These monitoring tools and applications do not provide protection; they are tools intended to give you an indication that something has changed so that you can take some remediation or recovery action. If configured properly, and if you are not currently performing system administrative functions, then they tend to generate few false alarms and generally will give you an alert/alarm when certain types of suspicious/malicious activity is occurring on one of your devices, systems, or computers. They cannot monitor and detect every kind of attack or exploit, just those that make, or result in making, changes to critical system policy and configuration settings. Fortunately, that covers a lot of ground. From a defense-in-depth standpoint, since each type of monitoring and detection technology we have described has some level of blind spot, combining them provides overlapping coverage and better cybersecurity. In fact, you will want to use these monitoring tools to monitor each other as well as your SCADA system assets. But then you will have multiple detection/prevention systems all telling you about things they have identified, so how do you bring all of this information together to get a complete picture? That is where SIEM technology comes into play.

Security information event management (SIEM)

Security information and event management (SIEM) solutions are a combination of the formerly disparate product categories of SIM (security information management) and SEM (security event management—also called log analysis). SIEM technology is supposed to provide near real-time analysis of security alerts (Syslog messages and Windows log entries) generated by network devices (switches, data-diodes), computers, security appliances (NIDS, firewalls), and even applications (including HIDS and AV/endpoint protection applications). SIEM solutions come as software, appliances, or managed services, and may also be implemented to log security data and generate reports for regulatory/industry compliance purposes (e.g., NERC CIPs or 10 CFR 73.54). SIEM products support the capabilities of gathering, analyzing, and presenting information from a range of systems and network and security devices, as well as from other security subsystems such as a HIDS and/or NIDS management console or endpoint protection applications. A SIEM may need to interface with the following:

- firewalls, NAC appliances, switches, HIDS and NIDS servers/consoles;
- identity and access management applications (RADIUS, Microsoft AD, etc.);
- vulnerability management and policy compliance tools; and
- operating system, database, and application logs (with third-party log forwarder)

A key focus of SIEM is to monitor and help manage user and service privileges, directory services, and other system configuration changes—as well as providing log auditing and review and incident response. SIEM products generally offer the following functions and capabilities:

- **Data Aggregation:** SIEM/LM (log management) solutions aggregate data from many sources, including network, security, servers, databases, and applications, providing the ability to consolidate monitored data to help avoid missing crucial events. Microsoft does not support Syslog functionality, so SIEM vendors often provide add-in application software to forward the various Windows logs (e.g., system, security, event, and application) to their SIEM. The SIEM has parsing logic to extract individual information items from logs and store that data in a time-correlated, relational manner.
- **Correlation:** looks for common attributes, and links events together into meaningful bundles. This technology provides the ability to perform a variety of correlation techniques to integrate different sources, in order to turn raw data into useful information. A SIEM will have user-configurable rules that direct what sorts of events it is looking for (e.g., log entries with designated Windows event codes), as well as vendor-supplied rules. The detection ability of the SIEM will be directly proportionable to the quality and completeness of the rule set defined and the data sources. NOTE: correlation of information in logs from disparate systems/devices requires that all of them be time synchronized. We will return to this point later in this chapter.
- **Alerting:** the automated analysis of correlated events and production of alerts, to notify recipients of immediate issues, usually with a severity ranking.
- **Dashboards:** SIEM/LM tools take event data and turn it into informational charts and displays to assist users in correlating and understanding the collected data.
- **Compliance:** SIEM applications can be employed to automate the gathering of compliance data, producing reports that adapt to existing security, governance, and auditing processes.
- **Retention:** SIEM/SIM solutions employ long-term storage of historical data to facilitate correlation of data over time and to provide the retention necessary for compliance requirements. Because the database can be quite huge and can require extensive processing to manage, some SIEM products use a *Hadoop/NoSQL* structure for their database and allocate many terabytes of storage for retention purposes.

SIEM platforms help aggregate logging and event data from distributed network-connected devices into a centralized database, store it, monitor it, report on it, purge it when the time comes, and provide the situational awareness necessary

to effectively respond to cyber events (fig. 11–16). Many of the popular SIEM products run on the Linux operating system platform (e.g., SE [security enhanced] Linux with extensive hardening) and use a multi-processor blade server design to get the necessary computing throughput. In this case, it is often possible to have the Linux O.S., on which the SIEM application is executing, send its own Syslog messages to the SIEM application so that the SIEM is somewhat self-monitoring.

Typically, configuring the initial log/Syslog message data parsing is an issue in implementing a SIEM, as is keeping up with the changes by Microsoft as they modify their log formats and add new logs. Although Syslog protocol is standardized, the actual content is vendor-specific, and so a SIEM product needs to have parser tools that extract information from the Syslog messages coming from various vendors' devices and subsystems. Also, for proper correlation of data from multiple sources, as previously mentioned, it is essential that all such sources be sufficiently time-synchronized.

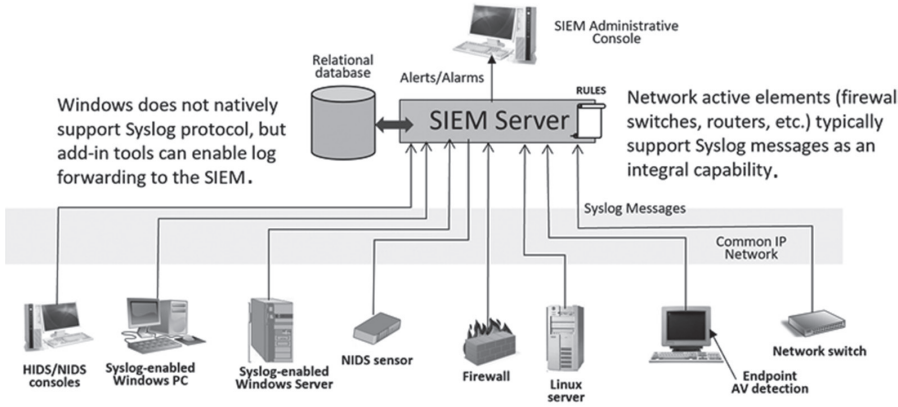


Fig. 11–16. Simplified block-diagram of a SIEM

SIEM products generally have sophisticated language structures for rule definitions and associated data dictionaries that define various types and classes of information. Rule definition languages and tools vary from vendor to vendor, but most have basic building blocks and a syntax for conditional checking. Building blocks can also be statistical calculations and data samples taken over time and even event counters. Most SIEM products also support conditional rules that can be activated or deactivated by other rules. SIEM rules tend to be much more complicated than firewall rules because they deal with many more types of information and data. For example, the following is a Windows log entry generated by a privileged (Admin) user login. There is a lot of information here, and the SIEM will have to parse this log entry into individual data elements and then,

presumably, there ought to be a rule that makes a check on Event ID Code 4672 to confirm that user wtshaw4 is an authorized administrative account holder and whether he is already/currently logged in. Syslog messages have a few mandatory elements (date, time, source IP, priority/facility), but the major part of the message is just text that the product manufacturer thinks provides sufficient detail to describe the event being logged. Microsoft has been guilty of changing their log format in different releases. The only limit is that a message cannot exceed 1024 bytes, including the mandatory data. The SIEM needs to be able to parse these messages and extract the relevant data:

> Jul 28 13:05:20 100.110.81.40 SERVER2 AgentDevice=WindowsLog AgentLog File=Security PluginVersion=7.2.3.1018506 Source=Microsoft-Windows-Security -Auditing Computer=SERVER2.NS.LOCAL OriginatingComputer=100.110.21 .23 User= Domain= EventID=4672 EventIDCode=4672 EventType=8 Event Category=12548 RecordNumber=44745096 TimeGenerated=1532808301 Time Written=1532808301 Level=0 Keywords=0 Task=0 Opcode=0 Message=Special privileges assigned to new logon. Subject: Security ID: Account Name: wtshaw4 Account Domain: NS Logon ID: 0x1c71008a6 Privileges: SeSecurityPrivilege SeBackupPrivilege SeRestorePrivilege SeTakeOwnershipPrivilege SeDebug Privilege SeSystemEnvironmentPrivilege SeLoadDriverPrivilege SeImpersonate Privilege SeEnableDelegationPrivilege

As was mentioned, the syntax and structure of SIEM rules is vendor-specific, but all products allow you to have rules with a string of testable conditions and values so that multiple factors can be evaluated by a rule. A popular and powerful SIEM product is IBM's QRadar© offering, which supports a sophisticated rule definition language. The following rule, in QRadar© syntax, examines the entries of the system security event log of a Windows computer looking for indications of an unapproved USB device being inserted into a USB port on any of the Windows computers. As we said above, the actual log entries generated by Windows are text and contain a lot of data, but not always all that much useful information. The SIEM rules need to be able to identify the particular data elements to check and what to look for—SIEM rules allow for the definition of objects and data fields—in this case, searching for the string "USBSTOR" anywhere in a log message from the security event logs from any of the Windows computers. If found, the alert generated is a "Windows Unauthorized USB Device" alert, which defines the actual text to present, the priority of the alert, and where the alert should be sent (to whom):

> Rule 110: **Windows Unauthorized USB Device** on events which are detected by the Local system **and when** the events were detected by **one or more** of Microsoft Windows Security Event Log **and when** the **Event Payload** contains **USBSTOR** and USBSerialNumber **is not Whitelisted**

The more time you spend developing comprehensive rules for your SIEM, the better it will be able to correctly identify questionable, unauthorized, and malicious activities. Also, the better and more detailed your rules, the less effort you will need, when an alert is generated, to make a determination of what the alert means and how to respond. If you have a rule that just says: "tell me if the NIDS sees something," then when you get that alert, you will have to go to the NIDS subsystem and dig through its records to determine the root cause of that alert. But if you have rules that say: "tell me if known malware is seen on the network," "tell me if known exploit message sequences are being seen on the network," and "tell me if there is any attempt to make an outgoing connection to an external IP address," then you will get alerts that are far more informative. As with our discussion of HIDS and NIDS, it is quite likely that a newly implemented SIEM will have issues with both false positives and false negatives as you work to add or modify rules and adjust the data parsing algorithms. It is not uncommon to spend several months tuning, adjusting, and modifying rules in order to get your SIEM system working the way you want.

A SIEM (and NIDS/HIDS) are like a burglar alarm; they let you know that something is happening and that you need to respond. If no one hears the alarm going off, then the alarm is purposeless. Similarly, this means that a SIEM requires that someone receive and review its alerts and be prepared to take appropriate actions. Most (if not all) SIEM products can send email and text messages when alerts are generated. That sounds good, but that means needing a way to communicate with email exchange agents and the Internet, which would put your SCADA system at risk by creating a potential attack pathway. We will discuss architectural strategies and technologies that can address this issue shortly. But first, let us return to the discussion about time synchronization.

Time synchronization

In order to combine messages and logs from a multitude of devices and computers, each with a local, independent time-keeping capability (a local hardware clock), there needs to be time synchronization among these network-connected devices, computers, and systems. This means having a mechanism that allows a periodic adjustment of any accumulated time drift so as to keep all systems, computers, and devices within an acceptable time window (usually to plus/minus one second or better.) A SIEM uses time tags on data to correlate events that may have been generated by the same activities. If various computers and devices have clocks that are drifting apart, then time correlation, and sequence of event determination, may be nearly impossible. Devices sharing a common IP-based network can make use of one of the various protocols for providing time synchronization, such as (S)NTP—the (Simple) Network Time Protocol. SNTP is not a secure protocol (and some server implementations have serious vulnerabilities), but on an isolated network

with its own precision time source (and monitoring via NIDS), that should not be a major concern. There are security extensions for NTP (so called NTPsec), but most implementations are not the secure version. NTP message spoofing attacks are often used to allow time-based token attacks by repeatedly resetting the time so all token variations can be attempted without the 60-second time window expiring. (An attack that should be detected by a well implemented NIDS!). Within your SCADA system, and its various IP-network-connected subsystems, you can provide time synchronization by using an accurate time source (today the most common being a GPS time receiver) and connecting it to a server that will run an SNTP service (fig. 11–17). There are also integrated GPS/SNTP servers that merely need to be connected to power an antenna/receiver and your local LAN. Then, in each computer, device, and subsystem, you must set up the SNTP client software. Most network devices, such as firewalls, routers, and switches, support SNTP clients, as do both Windows and Linux. There are accurate NTP time sources on the Internet, but as we have stated several times, you really mustn't connect your SCADA system to the Internet. Better to use a GPS time receiver and not introduce a possible attack pathway. If you have IP-networking to the field, then you could provide SNTP services to field devices over this network connectivity, if any require this. As has been discussed, in some industries, reasonable time synchronization of field sites is important for applications such as pipeline leak detection. If all of the devices, network components, security subsystems and computers sending logs/Syslog messages to the SIEM are receiving SNTP updates then this should provide adequate time synchronization. Note that if you are using firewalls to provide network segmentation then be sure to allow SNTP through those firewalls!

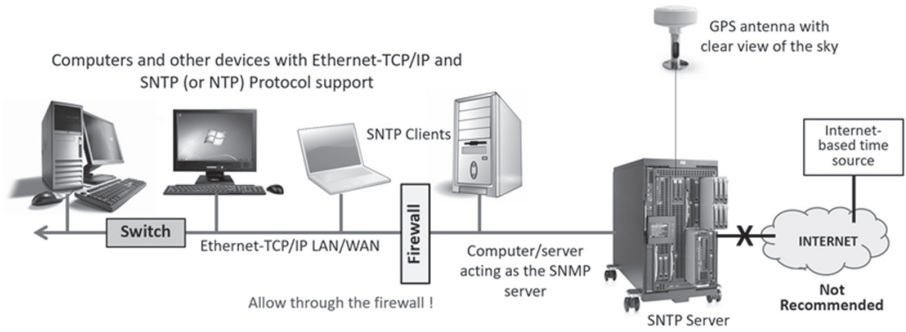


Fig. 11–17. SNTP server on the LAN, with GPS time source

Data diode technology

One way in which you can provide connectivity from your SCADA system to both corporate assets and corporate applications is to use *data-diode* technology

to create a secure, unidirectional (simplex) communication channel. Another is to establish a so-called DMZ and position shared assets within that isolated LAN segment. A data diode is called that because like a diode (which allows current to flow in only one direction), information/bits can only flow in one direction when using a data diode. This simplex operation is ensured in a physical manner, typically by using a single optical fiber as the transmission medium for sending data bits from one special-purpose computer module to another (fig. 11–18). A data-diode is a microprocessor-based communication device that creates a simulated bidirectional TCP session while actually providing a simplex channel that, by definition, transmits data/bits in only one direction. Diodes are used to send information from critical systems to untrusted systems, while eliminating the possibility of a cyberattack being initiated via that communication channel from the untrusted systems. UDP messages and Ethernet link-layer messages can also be sent through the diode, as long as no reply is needed or expected by the message sender. These are not plug-and-play devices. A great deal of diode configuration is required to establish system/application to system/application data exchanges. But a SCADA system could send data out to corporate systems and applications (and logs and Syslog messages to a SIEM) positioned on the corporate business LAN/WAN, with no risk of this creating an attack pathway, using a data diode (fig. 11–18). In this manner, the SIEM would have access to the corporate email server and even out to the Internet (through the corporate Enterprise firewalls) if that were necessary. And corporate applications requiring data from the SCADA system could get that data safely as well (but they could not send data back to the SCADA system). Data diodes can also generate Syslog messages whenever they are being configured or their settings adjusted if the receiving end fails to receive any traffic within a defined time window. So those could also go to the SIEM.

A major feature of data diodes is that they can simulate the actions of a TCP bi-directional session so that existing applications do not need to be totally rewritten to work in a simplex manner. TCP responses to the inside application are faked by the upstream single-board computer—based on receiving a message from that SBC, the downstream SBC initiates (or continues) a TCP session with the outside application (and responds to its messages). Making all this work takes some fancy footwork in the data diode.

Data diodes have been used extensively to protect critical government and military computer systems that contain classified and sensitive information, but, more recently, they have also found their way into industrial applications such as isolating/protecting SCADA and plant DCS systems.

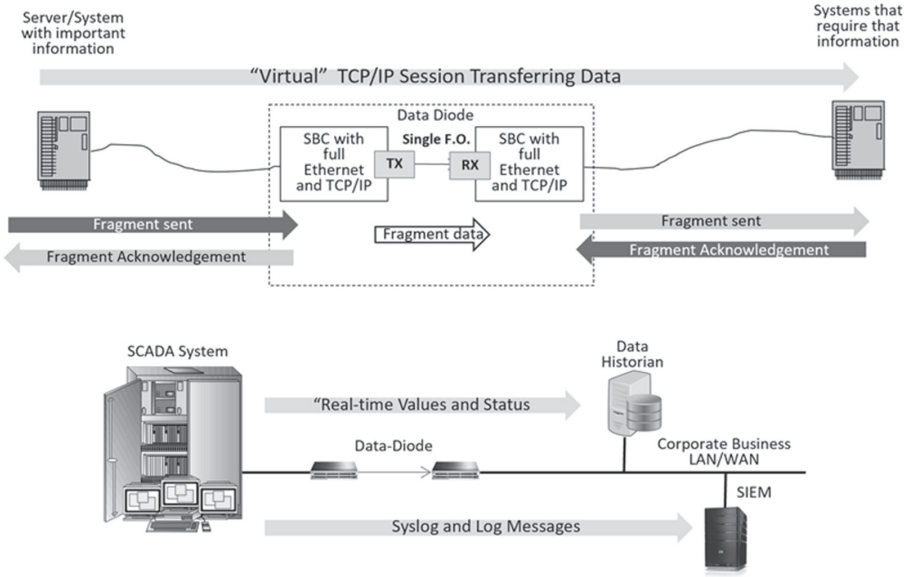


Fig. 11–18. Using a data diode to forward data safely

Establishing isolation via DMZ

In lieu of using data diode technology, a more well-known and conventional IT strategy for isolating critical systems (such as your SCADA system), while still permitting the flow of essential information, is to construct a *Demilitarized Zone* or DMZ (the term borrowed from the Korean war). A DMZ is an isolated LAN segment, positioned between a trusted network and an untrusted network, containing special systems that need to be accessible from both networks while NOT permitting direct traffic between the two networks. A DMZ can be created in a couple of ways, one being a single firewall (or router) with at least three interfaces. Another way is to employ two separate firewalls, one on each "end" of the DMZ network segment. This second approach (two firewalls) is considered as more secure, particularly if different manufacturer's firewalls are used (not just two different models from the same manufacturer), because a compromise of the firewall facing the untrusted network (e.g., the Internet) does not immediately mean that an attacker has gained access into the trusted network since they would still need to defeat the second firewall, and a different means of compromise will probably be needed due to the different type of firewall being used (fig. 11–19). In a DMZ arrangement, messages arriving from the untrusted network must have a destination IP address for a computer within the DMZ; otherwise the message is dropped. The same is true for messages arriving from the trusted network. Thus, the only way a message can move between the trusted and untrusted networks is if a computer in the DMZ forwards that message. But systems on both networks can communicate with the

computer(s) in the DMZ. If you place a DMZ between the corporate network and the SCADA network, then you could place the SIEM and a data historian in the DMZ (and probably a NIDS sensor as well). With this type of DMZ architecture, the SCADA system can send log/Syslog messages to the SIEM located in the DMZ, and the SIEM can communicate with the corporate email server (fig. 11–19). The SCADA system can read and write data from/to the data historian positioned in the DMZ, and corporate applications can do the same. But an attacker on the corporate network would not be able to send exploits to the SCADA system. They would have to attack and compromise at least one of the computers in the DMZ and use that to attack the SCADA system. For that reason, the systems in the DMZ ought to be hardened, including installing a whitelisting HIDS, and the message traffic coming into the DMZ from the corporate network ought to be monitored by the NIDS.

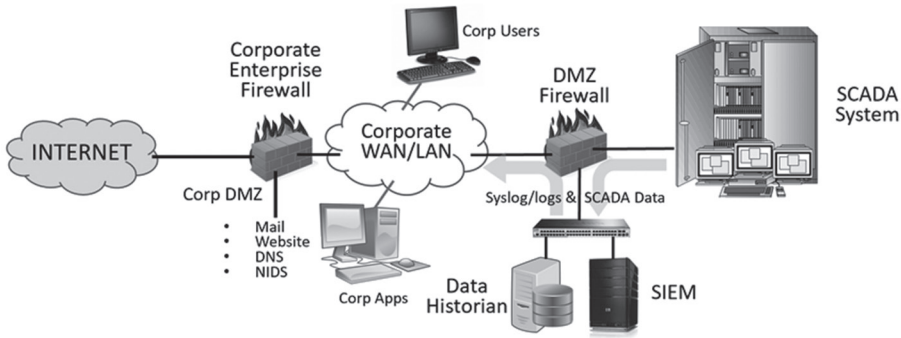


Fig. 11–19. Establishing a DMZ to isolate the SCADA system

It is very typical for a corporate IT organization to establish a DMZ between the Internet and their corporate WAN/LAN in order to allow systems such as the corporate Web server, DNS server, and email server to be accessible from the Internet as well as from the corporate network. It is also a good place to have a NIDS sensor to monitor incoming and outgoing message traffic. It must be understood that any system placed into the DMZ is visible and accessible to threat actors on the untrusted network (Internet or corporate WAN) and so needs to be given every possible cyber protection: patching, hardening, whitelisting, HIDS, etc. The International Society for Automation (ISA—https://www.isa.org) has a working group on IACS cybersecurity that highly recommends treating the corporate network as being untrusted because it has all too easily been compromised with social engineering ploys such as spear phishing, as well as placing a DMZ between your SCADA system network and the corporate network. As with the discussion of data diode technology, a data historian and the SIEM (hardened and whitelisted) can be positioned in that DMZ and thus be accessible to the corporate network (and have access to the email system and Internet), while providing a barrier against an attacker who has gained a foothold in the corporate network.

Ethernet switch security features

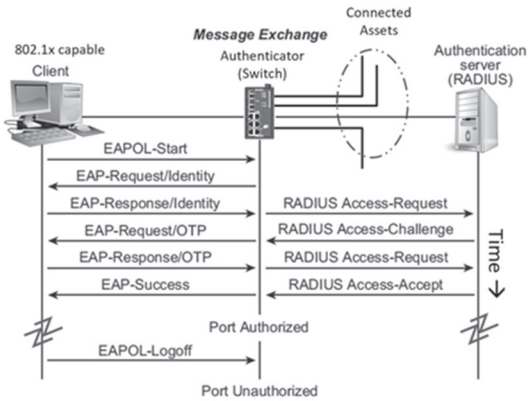
Ethernet switches are the foundation for your SCADA system LAN, and today, they have capabilities that go beyond merely passing frames around as quickly as possible. As we have mentioned, some switches can have firewall-type ACLs, some can support MACsec, most offer so-called port security plus VLAN support as well as logging with Syslog message generation, and the more advanced ones also support IEEE 802.11i switch-based NAC. What these features have in common is that they are used to block (or at least mitigate) unauthorized access to your SCADA LAN. Understandably, you probably assume that your SCADA LAN is physically protected and unassailable, that no attacker could gain physical access to make a connection anyway. The problem is that the events of 9/11/2001 proved that the threat actors are willing to go to great lengths to achieve their goals, and that could include penetrating your organization (getting a job there) or that of a trusted vendor that provides on-site or remote support (getting a job there), or compromising someone in your organization (money, sex, violence). And so, it just makes sense to layer as many protections as are possible onto your SCADA system (and they can protect against stupid human errors as well).

A simple security feature in most switches is port security, wherein you can assign one or more MAC addresses permitted to be connected to the particular port and the action to take if a device with another MAC is plugged in instead. With a few commands entered into its console port, a Cisco switch (for example) can be told to allow just one specific MAC address and to shut down and lock the port if some other device is plugged into that port (fig. 11–20). The Cisco IOS operating system supports an extensive command tree interface that can be accessed via the console port of the Cisco device (a switch in this case) using a serial COM: port on a laptop or PC, and which can be used for things such as setting up port security.

Fig. 11–20. Setting port security on a Cisco switch port

In a typical SCADA system, once you have everything set up and operating, there are typically not a lot of requirements to be connecting and disconnecting equipment. And it may be important that particular devices be plugged into specific switch ports (possibly due to the devices' QoS and VLAN settings on that port), and so configuring port security helps to enforce that requirement. Of course, it is possible to change the MAC address of most laptop PCs today, so in theory an attacker could change their computer's MAC to that accepted by the switch port, but first they would need to figure out what that value needs to be. A brute force attack (trying a range of MAC addresses) on the port would not be possible, as the first wrong MAC address provided will shut down the port. The attacker would need to get into the device authorized to be connected to the port and attempt to identify its MAC address. Also, unplugging a device from a port ought to generate a Syslog message to the SIEM and plugging in a new device ought to do the same and may also trigger a NIDS alert.

We have already mentioned the IEEE 802.1x port-based network access control capabilities of more advanced switches. Even if an attacker were to circumvent or overcome port security, having 802.1x running on the switch would place another roadblock in their way (remember defense in depth?) by preventing them from communicating with any other systems, computers, or devices on the LAN unless they could provide a valid user ID and password for authentication purposes (fig. 11–21). The IEEE 802.1x standard defines a protocol for client/server-based access control and authentication that prevents unauthorized clients from connecting to a LAN through switch ports. The purpose of the authentication server is to check each client that requests access to the port. The client is only allowed access to the port if the client's permission is authenticated. If you have a central authentication server (like Microsoft Active Directory) and devices/systems that support 802.1x security (like both Windows and Linux), then you probably want to implement 802.1x security on your switches. Figure 11–21 shows the message sequence between the supplicant (PC wanting access to the LAN), the authenticator (the switch the PC is plugged into), and the authentication server (the AD server). Once authenticated, the switch will forward Ethernet frames from the supplicant (PC) to any other of its ports, as needed, to deliver those frames. If the supplicant logs off or disconnects from the port, then the port is blocked again (unauthorized to forward frames) until the authentication process is repeated.



Three components are used to create an authentication mechanism based on 802.1X standards: Client/Supplicant, Authentication Server, and Authenticator:

Client/Supplicant: The end station that requests access to the LAN and switch services and responds to the requests from the switch.
Authentication server: The server that performs the actual authentication of the supplicant.
Authenticator: Edge switch or wireless access point that acts as a proxy between the supplicant and the authentication server, requesting identity information from the supplicant, verifying the information with the authentication server and relaying a response to the supplicant.

Fig. 11–21. Using IEEE 802.1x port-based NAC

Syslog message generation

Collecting and forwarding log messages, using Syslog protocol, may not seem like a cybersecurity measure, but if these Syslog messages are going to a SIEM, then they will be time-correlated with other log messages from other devices, computers, and systems and may help to indicate malicious activity. In addition, switches will send Syslog messages for a wide range of events and actions, such as connecting to and disconnecting from a port, making switch setting changes, logging into the switch, successful and unsuccessful 802.1x authentication events, and even setting time to the switch (fig. 11–22). Wouldn't it be handy to get a message if an Ethernet cable were accidentally disconnected or broke due to mechanical stress or that the LAN-connected report printer was out of paper? Windows does not natively support a Syslog server (an application to receive, collect, and present Syslog messages), but there are several free and for-purchase third-party Syslog server applications that can be installed on Windows. SolarWinds™ previously supported a limited-function, free Syslog server called Kiwi® (see fig. 11–22) that could be downloaded and used to test and experiment with Syslog functionality, and there are still free limited functionality Syslog server applications available. When used properly, Syslog functionality alerts you when something has changed or occurred that you may need to investigate or to which you need to respond. Most (if not all) network-connected smart devices (except those running Windows) support Syslog messaging (third-party add-ins for Windows can give it Syslog generation capabilities). As was mentioned, Syslog messages are time-tagged in the initiating device, and so maintaining reasonable time synchronization across your network is important.

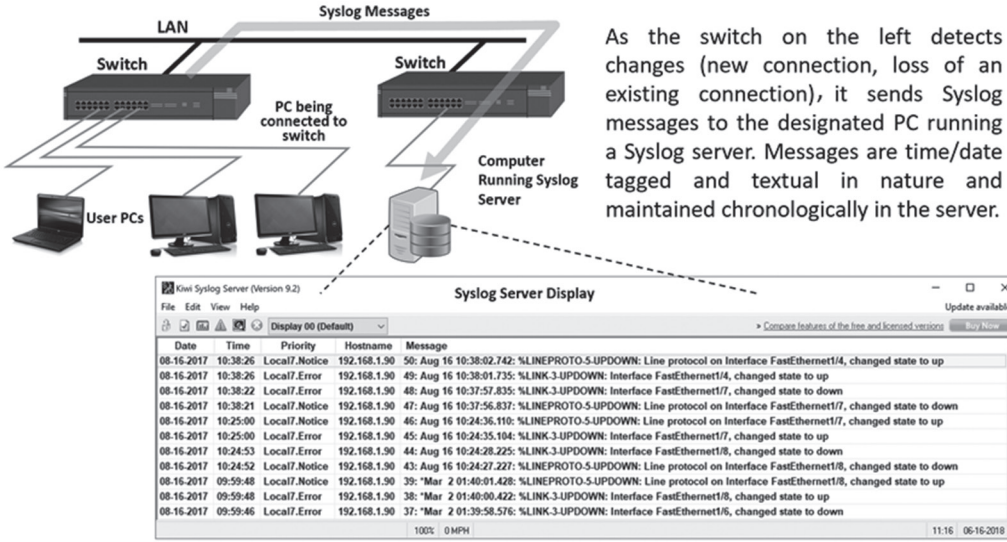


Fig. 11–22. Using Syslog messages to monitor network elements

Computer hardening

One final security strategy that can be applied to enhance your SCADA system cybersecurity is by performing *hardening* on your computers in order to minimize their *attack surface*. Hardening consists of several activities including: removing or disabling services, utilities, tools (e.g., program compilers/linkers), and applications that are not essential to the mission so that, in case they contain exploitable vulnerabilities (possibly yet unknown), those vulnerabilities will no longer be present. Hardening of computers is needed because the default factory configuration provided for most commercial operating systems is non-secure and full of known exploitable vulnerabilities. Windows, Linux, and Apple OS-X configurations usually start with everything enabled and loaded to accommodate inexpert users who just want to "use" the computer without any hassle and who desire (and truly need) "plug and play" support for any peripherals they add and any programs they install. Windows is especially filled with lots of goodies most users never need/want, many of which offer vulnerabilities to attack. Unfortunately, many IT departments (and SCADA vendors) often use "out of the box" distributions of a commercial off-the-shelf (COTS) O.S. and may not (generally don't) adequately harden them as part of their initial configuration. Hardening is a trade-off between usability and security. Some features, such as file and printer sharing via SMB messaging and DCOM, are well-known security vulnerabilities, but you may need them for your SCADA system to function. Hardening basically involves the following sorts of actions:

- Installing (after testing, if possible) all of the available security patches and updates for the O.S. and necessary applications
- Eliminating or disabling unnecessary system components and services
- Disabling dangerous functions, features, and unnecessary peripherals
- Closing down unnecessary TCP/UDP "ports"
- Installing a whitelisting HIDS
- Installing and configuring endpoint protection/AV/firewall software
- Eliminating unnecessary accounts and establishing strong passwords
- Setting Internet/Browser security settings and policies (if a browser is needed for SCADA)

The end result of hardening should be a computer/PC that offers few (if any) vulnerabilities for an attacker to prey upon. Serious hardening takes serious expertise and is system-specific (hardening Linux is conceptually the same but practically quite different from hardening Windows). The following are typical recommendations for hardening a Windows XP computer/PC. They vary in importance and utility, and some may not be compatible with your particular system configuration or environment:

- Disable Autorun on CD/DVD drives and USB devices
- Disable Dump file auto-creation (it can contain sensitive information)
- Disable the FAX/telephony facility, unless actually needed
- Delete the computer games and the Xbox support services
- Disable NetLogon (unless Windows Domain central authentication is being used)
- Disable NetMeeting and Remote Desktop sharing (unless needed)
- Disable Remote Desktop Help Session Manager
- Disable Remote Registry editing (unless central/remote administration support is required)
- Disable the auto preview pane in IE (don't "run" scripts embedded in documents)
- Disable Universal Plug & Play (UPnP) Device Host (stops driver loading and installation)
- Disable the Telephony API (TAPI) unless required
- Set the Wireless Zero configuration to Manual (don't allow automatic Wi-Fi connections)
- Disable Terminal Services (unless you need remote desktop or remote assistance)
- Disable simple (unauthenticated) file sharing
- Disable unnecessary peripheral devices and their drivers
- Enable and use the encryption of folder contents or the Encrypted File System

- Set up file permissions and use the NT file system (NTFS)
- For shared folders, set the user access number to a small value (like 1)
- Don't let ordinary users have administrative-level accounts
- Set up account lockouts on multiple failed login attempts
- Set up logging and auditing of login attempts/successes
- Disable the "Guest" account and all but one Admin account
- Set strong password enforcement on the PC (complexity and expiration/duration)
- Set up the "zone" protection settings in IE (if you require Web browsing on your SCADA)
- Close ports known to be attacked by malware, like TCP port 445
- Delete the automatic file associations for "scripting" file extensions (.vbs, .js, .pif)
- Enable Page File (VM page swap file) clearing on system shutdown
- Set up a "HOSTS" file rather than using DNS lookup to get necessary IP addresses (if any of your SCADA computers actually require name resolution services)
- Use static IP addresses for all components and computers rather than using DHCP
- Use a password-protected screensaver to lock the system when unattended (except on Operator Workstations/consoles, of course)
- Disable Last User Name display, but enable last login time/date display
- Disable NETBIOS Names Publishing (unless absolutely needed)
- Disable default Administrative File Shares (using a startup batch script)
- Disable DCOM services unless absolutely needed (many IACS require DCOM for OPC)
- Disable NULL Session support (always a danger if enabled)
- Modify default network settings for TCP/IP to reduce DoS vulnerability

Got all that done yet? Of course, there is an equivalent list of hardening tasks for a computer running Linux—and even some for you MAC owners! In truth, it takes serious expertise to harden a computer without breaking it and killing off something that you actually need. And some of the listed actions may not be viable with your particular SCADA software or system design.

One hardening activity that is not all that complicated is blocking TCP and UDP ports that are not needed for the computer's functions (which you may be able to determine by using Wireshark to capture traffic, if you don't know for sure). In a Windows O.S., you can select an interface and then drill down to the windowpane/panel (several clicks deep) that lets you set up port/protocol filtering (fig. 11–23).

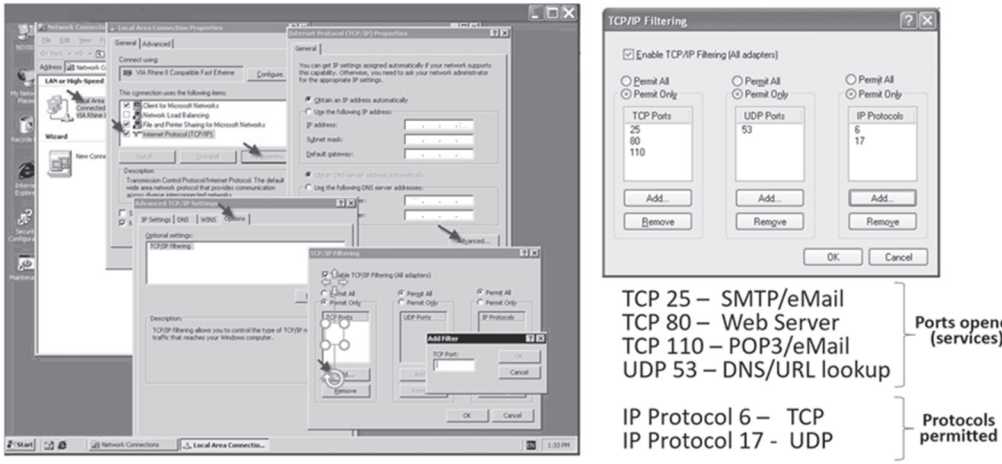


Fig. 11–23. Blocking unnecessary ports in Windows

Once you determine the services/ports and protocols you need enabled, you can use the Windows interface, as shown in Fig. 11–23, to enter the specific TCP and UDP ports (services) you will permit and also the IP protocols you will permit. This filtering feature can apply to all network interfaces on your computer if you check the check box at the top left of the screen. Otherwise, it only applies to the interface you selected to get to this windowpane/panel. This feature does not allow you to block ICMP traffic (since ICMP messages are required to make IP networking function). This feature does not affect outbound traffic using temporary/ephemeral ports (49152-65535 IANA or 1025-5000 in Windows) that are assigned in response to received connection requests. By default, Windows sets the "Permit All" option for all three categories. You can use the "Permit Only" option to specifically designate ports to be open, and Windows will block all others. Blocking IP protocols is a bit trickier since there may be some that might not occur to you, but which will be needed. We would suggest focusing on permitted/blocking TCP and UDP ports.

In many cases, hardening of a computer actually means setting the correct local policy settings for all of the hundreds of settings in a typical Windows or Linux system. Windows has a number of policy settings groups under the overall Security Settings heading in the Local Security Settings panel. Two examples of them, within a Windows XP/Pro operating system, are the User Rights Assignment group and the Security Options group. Each such group contains dozens of settings (fig. 11–24). Some settings are system-wide, and others are user-specific with a unique set for each user. (Note that a 'user' might be a program or process, not just an actual human being.) These "policies" are just a form of database with the content of a given record being the current setting value for the associated policy.

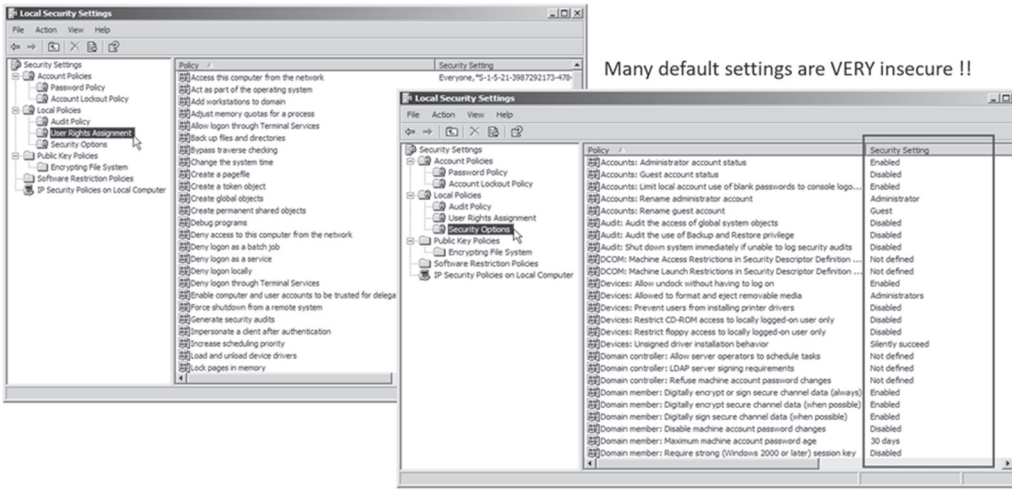


Fig. 11–24. Windows local security policy setting groups

By default, Windows comes with the most permissive set of configuration values. Various system functions and services check associated policy settings before performing a requested action by a user (or user-initiated process/program). You have significant ability to restrict user access rights and control sensitive functions by making the applicable changes to the policy settings. Of course, picking the proper settings values can be a challenge. In Windows XP, a security template is a text-based file that contains values for security-relevant system settings representing a particular security configuration. Templates can be created and updated using the Security Templates *Microsoft Management Console* (MMC). Templates may be applied to a local computer or imported to a *Group Policy Object* or Group Policy Management Console, which facilitates the rapid deployment of security settings across a Windows environment (this is what Microsoft's Active Directory server uses to push-out policy settings when a user logs in to a computer on the network). A security template is merely an editable text file containing lists of policies (by their name), grouped by category (e.g., "[System Access]"), with each individual policy assigned a value (e.g., "MaximumPasswordAge = 30"). You can actually make a copy of such a file, edit the values with the Windows WordPad tool, and save it as a new policy. Microsoft's Active Directory (AD) server has the ability to provide centralized user authentication, but it can do more than just authenticate. AD can have a set of policy settings for each defined role/job, and when a user authenticates, AD can send a table of local policy settings (called a GPO—Group Policy Object) to the computer of the user, and these will be applied, thus setting the permission and user access to the pre-determined level appropriate for the job/role of this specific user. That way, when Joe the technician logs into ANY SCADA system computer, his access rights will be automatically set to the technician settings.

Windows ships with several predefined security templates (fig. 11–25), but these may be inadequate or inappropriate for SCADA applications. Several organizations have developed and published their own templates, typically geared toward specific levels of security. Examples include the templates included with the Microsoft Windows XP Security Guide and the templates from the National Security Agency (NSA). Clearly, those NSA templates are designed for an environment containing classified, sensitive, Top Secret documents and thus probably would be too restrictive for use in a SCADA environment.

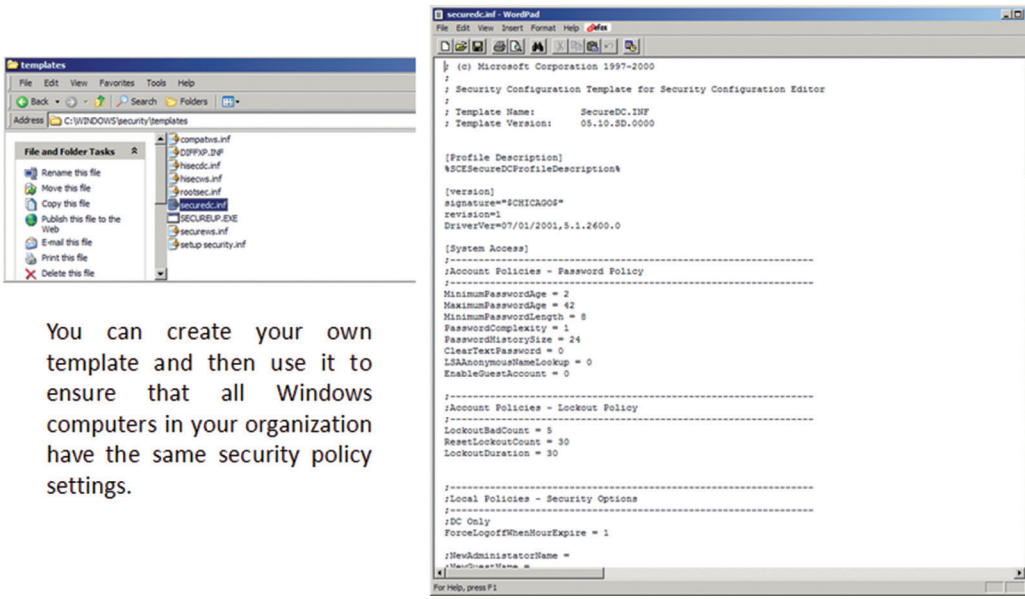


Fig. 11–25. Windows security policy templates

The point is that adjusting your security policy settings may be necessary to get a level of computer hardening that is adequate for your SCADA system cybersecurity. And one-size-fits-all is usually not a workable strategy when hardening. The hardening done for a server will differ from the hardening done for a laptop PC. And speaking of laptop PCs, if your organization uses them as test and diagnostic equipment then you need to take measures to keep them secure in case they are lost or stolen (a favored social engineering trick.) Enabling disk encryption (Bit Locker and EFS) will secure the contents and setting a BIOS password will further protect against an attacker trying to access the contents of the hard drive.

Hardening won't stop an attacker from trying to compromise your SCADA system, but it can remove most of the weak spots that the attacker will try to abuse, which may cause the attacker to give up—or at least force them to try more extreme tactics and, in doing so, make their activities easier to detect.

Virtual private networks

We have already generally discussed VPN technologies and the fact that a VPN allows interacting computers to authenticate to each other and use encryption to keep their communications secure. It also means that computers that are members of a VPN won't accept a TCP session or UDP messages from another computer if that computer cannot authenticate first. Further, it means that an attacker can't generate faked messages to any VPN member, because they cannot generate the proper encryption without having the session key being used, and they can't catch, alter, and then forward modified messages between VPN members (a 'man-in-the-middle' attack), because those messages include a message digest. Ok, but when and where should you apply VPN technology to your SCADA system?

VPNs are essential when the communications between and among the members need to traverse an untrusted network that offers an opportunity for attackers to position themselves to have access to the communications. Clearly, any communications from the SCADA system (or critical associated/support systems) that must traverse the Internet fall into that category. So, if your SCADA system has a persistent communications connection to an external party (regulator, regional authority, etc.) that uses the Internet as the communications channel, that connection really ought to be set up with site-to-site VPN protections. If you need to allow remote access on an occasional, temporary basis to a vendor or support organization, and that access is going to be via the Internet, then that temporary connection really ought to be protected with a mobile VPN gateway connection. But if you have leased (or private) point-to-point telecommunication circuits, then the use of VPN is less essential (although never a bad idea), since an attacker is far less likely to be able to gain access to that communication channel. One possible exception to this would seem to be the case of IP networking to the field because in this case, there is a greater possibility of an attacker being able to gain physical access to the communication channel at a field site. But in that case, the use of VPN really would not help. VPN technology does not prevent malware or an attack from being sent across the VPN-protected circuit, but it would potentially prevent a firewall or NIDS from being able to detect those activities (fig. 11–26). If an attacker got into a field site, they would just need to compromise one of the computers (or hook up their own—another reason for setting port security) and use it as a platform for attacking the SCADA system. A VPN connection would do nothing to block such traffic. There have been unfortunate cases of poorly positioned NIDS sensors, which is why one of the recommendations in NIST SP-800-53 (https://nvd.nist.gov/800-53) is to ensure that content-checking mechanisms (e.g., a NIDS sensor) are placed prior to or after encryption/decryption mechanisms. Because several forms of IP communications make use of SSL (Secure Socket Layer) encryption it is common for devices such as firewalls and NIDS sensors to support real time SSL decryption of message traffic but that does not extend to other forms of encryption. Just as a side note, NIST has published a range of recommendations on cybersecurity, and their special publications (SP 800) series is chock-full of quite good, diverse (and free) advice.

Even if you use a site-to-site (or mobile gateway) VPN, you still want to have a firewall and NIDS protecting that connection at the SCADA end, because you may not (probably don't) have any assurance of the cybersecurity and physical security at the other end of the connection. Laptop PCs have been stolen out of parked cars and hotel rooms and used to attack corporate IT systems, even though the connection between the laptop (now in the hands of an attacker) and the corporate systems was protected with VPN technology.

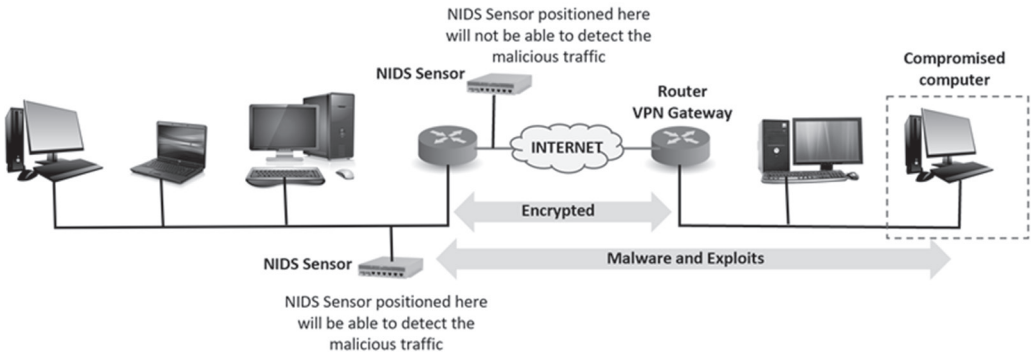


Fig. 11–26. Possible incorrect placement of NIDS sensor when VPN is used

Virtual LANs

VLAN technology in switches has been mentioned previously, but here we want to discuss why it should not be considered as a cybersecurity mechanism because some IT organizations believe that it is an effective means for blocking unauthorized access to systems and computers. VLAN technology is a switch-based function that causes switches to refuse to deliver Ethernet frames to destinations if the sender and destination do not have the same VLAN designation (numeric value). VLAN information is normally added to an Ethernet frame when that frame is received by the switch via an access/service port (not typically if arriving on a trunk port although some switches do that also if no tag is present in the arriving frame). The VLAN tag (a numeric value) is carried in the Ethernet frame as/if it moves between switches (over trunk connections), and at the final switch, the frame will be stripped of the tag and delivered out the appropriate access/service port. But none of that delivery stuff happens if the VLAN number assigned to the initial port differs from the VLAN number assigned to the final port. So, in theory, if an attacker were to plug into a switch on a port with VLAN 3, they could not attack a computer plugged into a port (on the same or a different switch) with a VLAN setting of 6. The problem is that many switches have an option (often the default setting) that allows a received frame that is already tagged to retain that tag (the alternate and better approach is to overwrite the tag VLAN value with the value actually assigned to the port). The reason for this is that people don't always

remember to set both ports used to connect two switches (one on each switch) to be trunk ports. So, a switch might tag a frame and send it out a trunk port and over the link to a second switch (where it is received on an access port), and rather than overwriting the legitimate tag values, the receiving switch just allows the existing tag to be retained. This opens up a vulnerability that hackers can abuse. There are hacking tools that allow the user to craft Ethernet frames (and the TCP/IP packets carried by the frames), including adding the VLAN and QoS tag values. This means that an attacker can plug into any switch port, regardless of the VLAN designated for that port, and send message traffic to computers and devices on the LAN, regardless of their LAN (fig. 11–27). There is nothing wrong with using VLANs to help segment and organize LAN communications, but be aware that this alone will not make your LAN cybersecure.

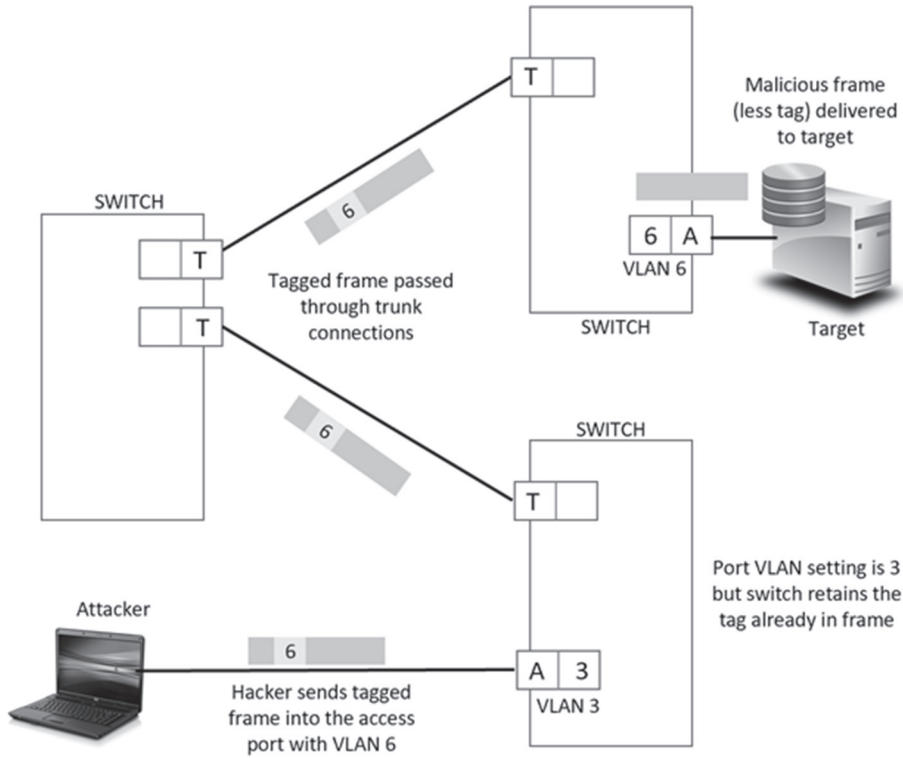


Fig. 11–27. Accessing VLANs using pre-tagged frames

Using HOST files to eliminate DNS

In most instances, it is unlikely that your SCADA system will require DNS (domain name service) computer name lookup services, as that is a function typically required for email and Web browsing on the Internet. Unfortunately, the

DNS system has known exploitable weaknesses and normally requires Internet access (to resolve external domain names); thus, it should be avoided if possible. But a DNS server can be configured for an isolated system just to permit name to IP address translation, and some SCADA systems may need DNS for some of the cybersecurity functions, such as having a SIEM or a firewall send an email alert (although you may be able to provide an IP address in place of a URL). Also, from an engineering and maintenance standpoint, it is sometimes cumbersome to have to remember IP addresses for all of your computers, switches, firewalls, and network-connected devices if you need to open up a telnet, ssh, VNC, or remote desktop session. It can be convenient to be able to give computers descriptive names and use these names rather than IP addresses. In the early days of the Internet, all computers on the Internet (all dozen or so) had a file called "hosts" in which a textual list of computer names and their associate IP addresses was maintained. Whenever a new computer was added to the Internet, everybody added them to their host file. Clearly, that strategy did not scale well and would be useless today. So the gods of the Internet devised a new scheme: servers on the Internet that would contain all of that information, and a naming scheme that allowed people to register/reserve a unique "domain name" that would be linked to their IP address. This is what DNS service is, and in most corporations, the IT department will be running their own DNS server to manage their local sub domains (e.g., sales.bigcompany.org, fieldservice.bigcompany.org, engineering.bigcompany.org, mail.bigcompany.org, and, of course, www.bigcompany.org). If, for some reason, your SCADA system needs DNS service, then in theory it could query the corporate DNS server, but that would mean needing a communications pathway into the corporate network, which we have already decided was not a good or secure idea. In most instances, your SCADA system computers and devices will have static, private IP addresses that will not change. What would be handy is a means for taking those IP addresses, putting them in a list with descriptive names, and then having the computers refer to that list when needed and not having to make a name-to-IP translation request to a DNS server. Oh yeah, like a HOSTS file!

With Windows, you assign a NetBIOS name to your computer as part of setting up the operating system. That name can be used among Windows computers and with various Microsoft tools and applications. But Linux has no support for NetBIOS names, and neither will your firewalls, routers, and non-Windows devices and systems. But, fortunately, that original scheme from the early days of the Internet of having a hosts file actually still exists and is supported by both Windows and Linux. In a Windows system, in the directory: %systemroot%\System32\drivers\etc (see Fig. 11–28) is a set of special files that are used by the operating system. In that directory, you will find a "hosts" file (no ".' extension). This file can be used to populate your DNS cache (without having to query a DNS server), and changes to the file automatically trigger a reload of file entries to your DNS cache. You can create a "hosts" file in each Windows/Unix/Linux computer, containing a full list of all your SCADA system computers and devices, giving them a name and IP address. This is

how it was done pre-DNS and it still works. Fig. 11–28 shows the default "hosts" file from a Windows 10 computer showing the definitions for "localhost" in both IPv4 and IPv6. Localhost is the name you can use for local loopback purposes. You can put a name and IP address entry for each of your computers, network elements, and network-connected devices and then use these names rather than remembering the IP address. Host files work in both Windows and Linux systems, but a similar file named "lmhosts" is in the same Windows directory and is just for Windows NetBIOS names (the name you assigned to the computer when you initially configured Windows). The "lmhosts" file is in the same directory and used for defining NetBIOS names for computers and their associated IP address. This file can also be used to designate a host as the domain controller in an MS Windows domain. You can also specify that some/all of these names and addresses be loaded into the local NetBIOS name cache at system startup. This is specific to MS Windows and not part of the Internet domain name scheme (the ***hosts*** file is for that purpose). If Windows Internet Name Service (WINS) is not available on the network, the "lmhosts" file can be used to support the subnets that *do not have a WINS* server, and to provide a backup name resolution service in case the WINS server is not available. (We mention this because older Windows-based SCADA systems may require a Windows domain control and WINS server to function.)

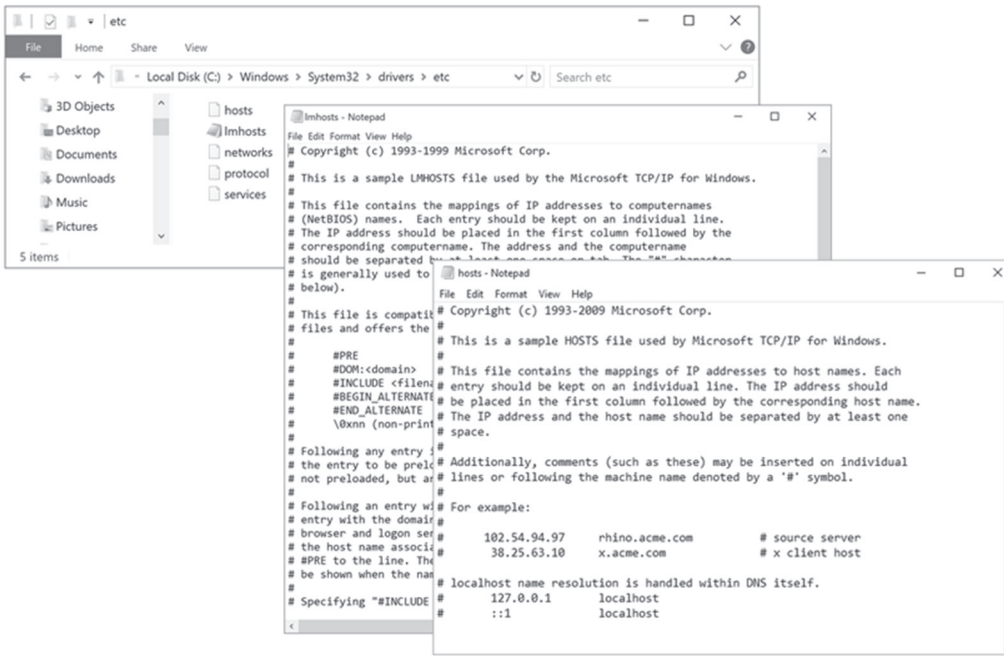


Fig. 11–28. The "hosts" and "lmhosts" files in Windows

Using KVM switches

In a SCADA system control room, it is typical to place a lot of computer screens in front of each operator so that they have a good view of the process and a means for setting up multiple information views to aid in performing their duties. Generally, there is no reason an operator ought to be exchanging emails with other parties or browsing the World Wide Web while on duty. If this capability were needed, then a separate PC, connected to the corporate network, would be placed somewhere in the control room. But it would not be part of the SCADA system, because those functions should not be possible via the SCADA system (or a really dangerous attack pathway probably exists).

On the other hand, a SCADA system engineer may need to frequently switch back and forth between the SCADA system and engineering support and business systems on the corporate network (including Web browsing and sending email) in order to perform their functions. In a prior chapter, we discussed why the solution should NOT be putting two Ethernet NICs in the engineer's PC and connecting it to both LANs (a so-called multi-homed configuration). But this could be SAFELY accomplished by putting multiple PCs, each connected to a different LAN, on the operator's or engineer's desk and letting the operator or engineer jump around from screen/keyboard to screen/keyboard. But that takes a lot of space, so this can also be accomplished by putting just a single keyboard, display, and mouse on the person's desk and hiding those two PCs under the desk or in an adjacent room. This second alternative is often the preferred choice (because it leaves the poor engineer/operator some usable desk space), but it requires the use of a KVM (keyboard, video, mouse) switching device. KVM technology has been around for quite a while. In its initial form, it consisted of electromechanical (or just mechanical) switches that would connect the single PS/2 keyboard, mouse, and RGB signals to one of multiple PCs, while the other PCs had nothing connected to them. This became a problem when those peripherals became smart, especially the video displays, and when they converted to USB connectivity. Your computer's video circuitry interrogates the display and sets its resolution appropriately. The same goes for your keyboard and mouse. Your PC may go into a special "headless" mode if it detects that there is no keyboard connected. Gaining and then losing all three peripherals (keyboard/video/mouse) can potentially cause your PC to do a reboot. So modern high-end KVM switches (the USB kind) now provide fake signals to your PCs not actually connected to the peripherals, so that they stay happy and in the correct mode for when they do get the peripherals back. Neither the old original KVM switches or the USB kind create an attack pathway by bridging the separate PCs together in some manner. But some USB KVM switches provide extra USB ports into which you could plug a flash drive, and that drive is switched from PC to PC along with the three peripherals. This is the same thing as unplugging the flash drive from one PC and plugging it into the other PC. We have already discussed the need to manage and control portable media and to possibly restrict authorized/trusted portable media to the SCADA system. With

a KVM switch, there is the possibility of accidentally connecting trusted media to an untrusted system/network and/or connecting untrusted media to your SCADA system (fig. 11–29).

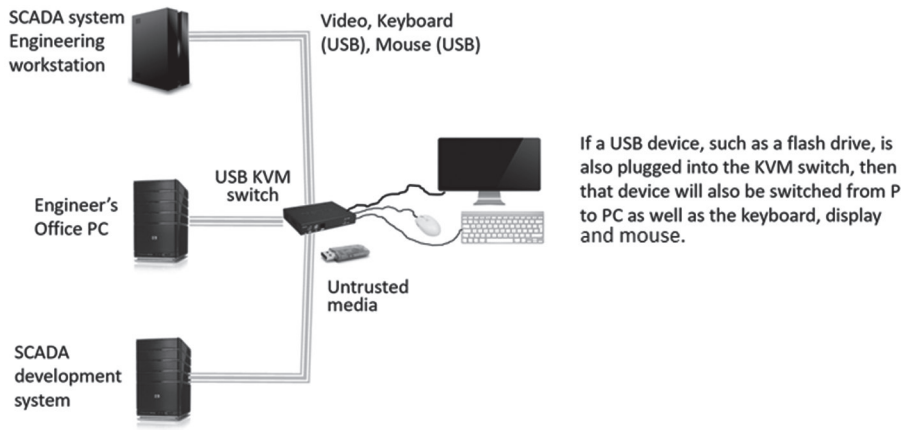


Fig. 11–29. Using a KVM to support multi-computer access

Using a KVM switch and multiple PCs is far more secure than allowing a multi-homed PC to be used, but with a USB-based KVM, it is recommended that USB ports not required for the basic three peripherals (monitor, keyboard, and mouse) be disabled in some manner, possibly through the use of physical port blockers. There are special high-security KVM switches designed to meet the cyber security requirements of three-letter government agencies. They actually incorporate a USB-based data diode design within the KVM switch. You probably don't need to go to that level of security in your SCADA system.

Remote Access Technology

As an alternative to a KVM switch it is also possible to make use of remote console support technologies such as Microsoft's Remote Desktop or the operating system independent Virtual Network Console (VNC) functionality that works for Windows as well as most Unix/Linux platforms. With these technologies you can pop up the desktop display of whichever computer you want to utilize and then just use your keyboard and mouse as if you were sitting in front of the other computer. It is even possible to have multiple such connections active concurrently and jump from compute to computer seamlessly. The downside to this is, of course, that setting up such remote access services in these various computers creates a potential vulnerability that could be used by an attacker were they to gain access to your network. It is possible to assign a password to these technologies and some of the VNC products support digital certificate-based authentication. If you use these technologies be sure to put strong authentication in place to prevent

them from being abused. And of course this only works for computers that are network accessible so it won't help with getting access to a computer on the business LAN as the SCADA system should not have such a connection, as we have already discussed.

12

Electric utility industry–specific cybersecurity issues

The electric utilities were one of the first to use computer-based supervisory control, and they continue to be a major proponent of it. Electric power transmission systems were being monitored and controlled by computer-based SCADA systems as far back as the early 1960s, when only very expensive mainframe computers were available. Part of the reason for this early adoption is that scheduling, optimizing, and controlling electric power generation and transmission requires very complex models (e.g., *state estimators*) and advanced applications (e.g., *automatic generation control*—AGC) that can't be performed, in anything approaching real time, except by rather powerful computers.

SCADA systems initially provided a way to keep track of and manipulate the available power being produced by an electric utility's various generating facilities, as well as a way to match that power with real-time demands over their large geographic service areas. As SCADA technology advanced in parallel with computer technology, the applications running on electric utility SCADA systems became more diverse. SCADA systems could keep track of historical power demands, over each service day, on the basis of what type of day was being experienced (in terms of weather, time of year, weekday vs. holiday or weekend, etc.). This information allowed electric utilities to predict the hour-by-hour power demand for the next day and to schedule their generation assets appropriately. In addition, regional SCADA systems were established to coordinate the various utilities and to manage the overall national power grid. This enabled utilities to buy and sell their generation capabilities to better match supply and demand.

Electric utilities have historically been—and continue to be—major purchasers of RTU equipment and have played a major role in developing many of the protocols still in use today. (Some legacy protocols are even named for the utility that sponsored the development, like PG&E protocol.) Electric utilities have industry-specific requirements both for their RTUs and for their advanced application programs. In Chapter 1, we explored some of these unique requirements.

Although electric utilities were one of the first adopters of SCADA technology, they have also been the most hesitant—or possibly just the least able—to integrate changes and adopt new technology (for a variety of operational and economic

reasons). Many electric utilities have continued to use obsolete legacy communications protocols because they have a huge installed base of old RTUs—and maybe even a SCADA system that may not be capable of being upgraded to support other protocols (because the vendor is long gone). There are RTU manufacturers who produce RTUs specifically for the electric utility market because of their unique requirements and legacy protocols. Electric utilities have also been a major user of leased analog telephone lines, because that is what many of the older RTUs were designed to utilize. It is not unusual for major electric utility SCADA systems to still be using legacy protocols running at 1,200 and 2,400 bps data rates (1.2 and 2.4 kbps, respectively) over several dozen analog leased telephone lines.

Furthermore, unlike the two other major industry segments that use SCADA technology (water/wastewater and pipeline industries), electric utilities have generally used only the most basic RTU capabilities (analog, pulse, and contact inputs and contact outputs), even though RTUs have progressed to become quite powerful and flexible. Nevertheless, over the past several decades, market forces and technology developments have driven the electric utilities to rethink their notions about communications and what comprises an RTU. In addition, as the cost of SCADA technology has dropped owing to the development of low-cost yet highly powerful PCs and servers, electric utilities have extended SCADA applications into their power distribution systems. This same market drive has also made SCADA systems available to small electric utilities (e.g., rural cooperatives and small municipalities) that previously couldn't afford to own and operate a SCADA system.

A major electric power transmission SCADA system can be quite a large and sophisticated system—with full automatic redundancy, several interconnections to other systems, and often a backup alternate operating site, where a (nearly) duplicate system waits, ready to take over in the event of a disaster (fig. 12–1). Electric utilities have tended in the past not to directly interface with the DCS systems that run their respective generating plants; rather, RTUs were typically placed on-site, and momentary contact outputs send ramping (raise/lower) commands to these systems and return a select set of real-time measurements. When they are interfaced, it is usually because the plant DCS system can provide an RTU emulation and communicate in the same serial RTU protocol used by the actual RTUs. In the 1990s, a major ISO in the western U.S. initiated a project to place what they called remote intelligent gateway devices (RIGs) at all of the generating facilities in their service area and to connect them to their two EMS operating facilities over the Internet. This may have been one of the first uses of the Internet for real-time monitoring and control.

It is very common for electric utility SCADA systems (or EMS systems) to have real-time communication connections with regional coordinating authorities (e.g., a power pool—or today, an ISO or even an RTO). These connections were usually via leased telephone circuits (possibly digital) and utilize either the ICCP protocol (discussed later in this section) or an RTU protocol. Today, those

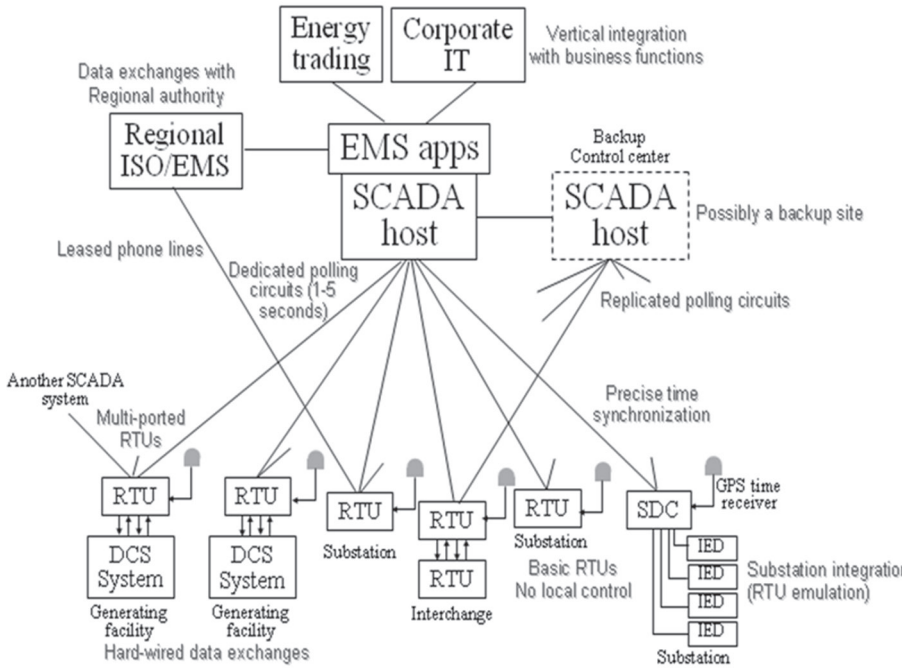


Fig. 12–1. Generalized block diagram of an electric utility SCADA system

connections might be via the Internet and protected from compromise using VPN technology. ICCP is an IP network-compatible protocol. For forecasting purposes, there may be data links to weather information suppliers, and for coordination purposes, there may be data links to the SCADA systems of adjacent electric utilities.

Unlike in other industries, things happen very quickly in the electrical world. A power transient can travel at nearly light speed and can trigger protective relays (to trip/close the circuit breakers they control) in several electrically adjacent substations in just small fractions of a second. Thus, in electric utility SCADA systems, it is more and more common, especially with the availability of low-cost GPS receivers, to have high-precision time synchronization (to the millisecond or better) and local time tagging of recorded events. Because conditions can change fairly rapidly, electric utility SCADA systems have tended to use individual polling circuits to each RTU, rather than using a multi-drop (multiple RTUs on a shared circuit) approach to reduce the number of polling circuits. This results in a system architecture with a large number of RTU polling circuits, as compared to a pipeline or water/wastewater system with a comparable number of remote units. Today, they might have broadband IP networking out to the major substations and other transmission facilities.

Geographic distribution of substations and generating facilities might also make line-sharing impractical. Electric utility SCADA systems tend to make use

of multi-ported RTUs to provide simultaneous access from multiple systems. In the past two decades, RTUs have begun to be replaced with *substation data concentrators* (SDCs), which emulate RTU protocols and collect real-time data from multiple sources within the substation.

Substation Backdoors

To understand the unique attributes of electric utility SCADA systems and the corresponding cybersecurity issues, it is necessary first to consider the main application of SCADA technology within electric utilities. The overwhelming majority of RTUs interfaced into an electric utility SCADA system will be located in electrical substations, for the purpose of controlling switch gear and transformers and making electrical measurements. Others will be located at generating facilities, for the purpose of issuing generation controls (to ramp generating units up or down) and making local electrical measurements. Still other RTUs will be located at key locations in the power transmission system, such as at interconnection points (*inter-ties*) with geographically adjacent utilities. (The majority of RTUs are in substations because there are usually a lot more substations than generating plants and system inter-ties.)

Although *generator ramping* is an important task, it is the control of switch gear and transformers that can have the greatest impact on the electric power grid. To a certain degree, the components that form the electric power grid are like a row of dominoes, and like a row of dominoes, if you set one falling, it can take down the next, which takes down the next, and so on. Protective relays and switch gear are positioned at various key points in the grid to break the chain and thereby limit damage (a process called *islanding*); the Northeast blackout in August 2003, however, demonstrated that this process doesn't always work as intended. Thus, it is vital for an attacker to be prevented from gaining communications access to substation RTUs, because that would provide the ability to control and operate switch gear and transformers. (Here we are talking about transmission-level substations, not distribution-level substations.)

In a traditional electrical transmission substation, circa 1990, the site RTU would monitor the status of circuit breakers, re-closers, switches, transformer tap changer position, capacitor banks, and other equipment (fig. 12–2). It also typically has momentary contact outputs for controlling and operating much of the same equipment. In addition, the key bus and feeder currents and voltages are brought into the RTU for transmission to the SCADA system.

In parallel with the RTU will be protective relays (and other devices) that make many of the same measurements and operate the same equipment, for the purpose of preventing severe damage to the plant equipment and the power grid. In the past, these protective relays were dumb electromechanical devices. But for the past three decades or so, they have become microprocessor-based devices (a.k.a. *digital*

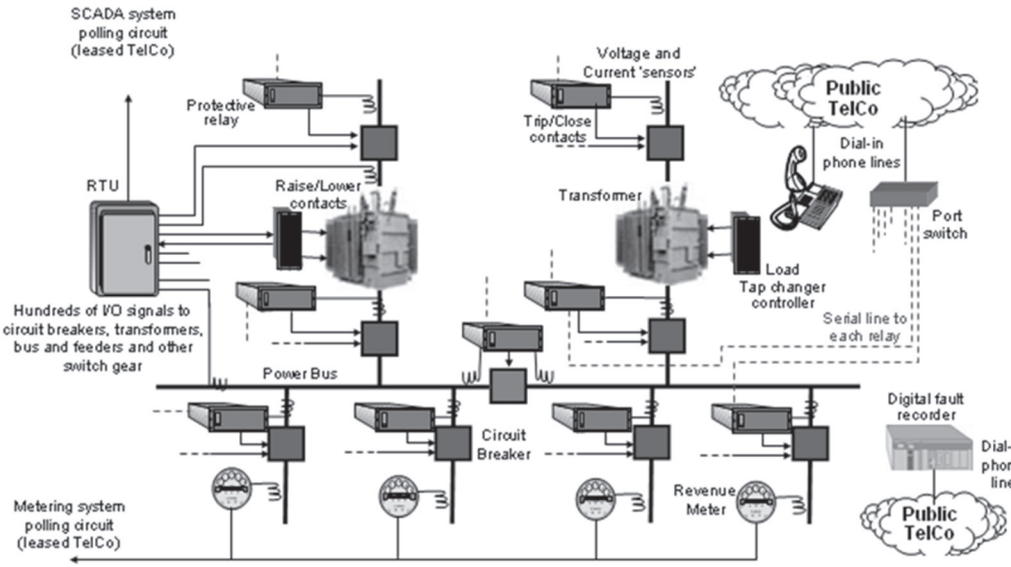


Fig. 12–2. Electrical transmission substation circa 1990

relays) capable of some level of both serial and Ethernet-TCP/IP communications. Modern digital relays usually support Modbus and DNP3 serial (and IP) protocols, as well as other TCP/IP protocols such as UCA2 with its Ethernet-based GOOSE messaging.

Electric utilities, particularly large ones, have often organized their internal personnel into groups responsible for different aspects of the operation:

- One group would be responsible for the protection of the system and equipment and would handle all of the protective relaying. This group would engineer, install, and configure the relays and would be responsible for examining the data collected by these relays, and other monitoring devices, when a fault occurred.
- Another group would be responsible for measuring (metering) the power/energy being bought and sold. This group would install and calibrate *revenue meters* and operate a separate data collection system designed specifically to collect meter data and compute billing information.
- A final group would handle the real-time operation of the transmission system. This group would own and operate the SCADA system and place the RTUs into the substations, inter-tie points, and other necessary sites.

It was common for substations to have numerous, parallel communication (telephone) circuits: one or more for each department or group—and even one for an actual telephone!

The primary danger if an attacker were to gain communications access to an RTU would be that the attacker could then operate critical switch gear—for example, a circuit breaker controlling a major power delivery circuit—or could trip a transformer, thus causing a domino to fall in the analogy invoked previously. To accomplish this, however, a knowledgeable attacker would not have to penetrate the SCADA system or wide area communications system. Most large substations have a backdoor that can be used for the same purpose (i.e., to control equipment). The protective relay groups in most large electric utilities want quick access to the data captured by relays whenever a fault occurs (e.g., a momentary outage due to a lightning strike) to identify the type of fault, to evaluate the potential damage to equipment, and to determine the effectiveness of the protection scheme.

Today, a typical protective relay is capable of continuously recording voltage and alternating-current (AC) waveforms on all three phases, over several dozen or more cycles. This information, along with precise data on contact event timing, provides tremendous insight into the causes and effects of faults and the efficacy of the protective schemes. Also, the logic and calculations performed in a relay, to determine when to take a protective action, can be adjusted by changing any number of settings. Often, for various reasons (e.g., expansion of the physical system or seasonal changes), these settings need to be adjusted.

Furthermore, since substations are distributed over large geographic areas, it is understandable that with digital (microprocessor-based) relays, the protection organizations would seek to establish remote communication mechanisms. In the past, this would have been accomplished by the placement of a *port switch* on an incoming dial-up telephone line and connection of the communication/configuration ports of all of the relays to this switch. With such a scheme, a relay engineer (or a hacker with a war dialer program) can dial into the desired substation from his desk, using a PC and a modem, and then send a short character sequence that causes the port switch to select the desired port/relay. Security is provided by having each relay require a password. Of course, most relays are set to the same password for convenience (often the factory default, e.g., "otter tail," which was used by a major relay manufacturer). If the passwords have been changed, it is common to use the name of the substation as the password. Most relays place no limit on the number of times you could try to log in and enter the password—which is a very convenient feature for an automated password cracker program.

If one is communicating to these relays, there is no need to bother with the SCADA system or RTUs, as the relays control the circuit breakers (and other switch gear) and can, depending on make, model, and configuration, be commanded to trip or close the breakers via the dial-in port. This dial-in relay connectivity is one of the greatest vulnerabilities in the electric power grid, and most utilities have taken steps to close this backdoor to their substations. Port switches have gotten smarter, and manufacturers have added levels of access control and authentication; some can now even perform encryption.

With changes in technology, the design of substations has also evolved, and most modern substations are now filled with intelligent electronic devices (IEDs). Many IEDs support some form of communications and also measure the same signals and values that are monitored by the RTUs. In an effort to reduce costs associated with instrumenting a substation, some utilities have implemented "*substation automation*" (a bit of a misnomer). This process entails changing the RTU from a device with lots of physical I/O to one with a lot of serial ports and little or no physical I/O. If a relay or power meter is already measuring the voltages and currents (probably with greater accuracy than the RTU), then why absorb the high costs for double-wiring those same measurements into the RTU, when a simple point-to-point serial communications link would allow the RTU to poll and fetch that data from the relay or meter?

Today, it is not uncommon for an RTU (or an SDC) to support 16 to 64 serial ports and a variety of IED protocols, just for the purpose of collecting and concentrating data from the various substation IEDs (like a small SCADA system within the substation; see fig. 12–3). In the vast majority of these instances, the data collected are normally only the simple, real-time data (e.g., real and reactive power, currents, and voltages). Many substation IEDs also contain a lot of other types of data (e.g., time-sampled AC waveforms and SOE logs) not typically collected by the RTU or the SDC. This is because the RTUs (and SCADA systems) usually don't support such exotic data. (Few, if any, SCADA systems have an application for viewing AC waveforms.) Moreover, many of the protocols used for communications between an RTU and a SCADA system generally don't support the transportation of such complex, compound data (except by playing tricks). For these reasons, most substations still require a separate communication pathway, so that protection engineers can access their relays and fault recorders, upload fault records, and change settings.

In some instances, the SDC may be substituted for the port switch, to offer a *pass-through mode,* whereby a dial-in connection to the concentrator would allow the caller to temporarily take over any one of the links to the various relays (or other IEDs). In effect, the SDC would act like a port switch for connections made via the separate dial-in telephone line. When the caller hangs up, the ports are reclaimed by the SDC, and polling of the IEDs resumed.

The use of a dial-in telephone line still presents a potential backdoor through which the substation could be attacked. This separate dial-in telephone line might be replaced by a network WAN connection in some instances, and the SDC might be capable of supporting some form of cybersecurity, such as link encryption or even full VPN with authentication. SDC technology has been around for more than two decades now, and the product capabilities and architectures vary widely from vendor to vendor.

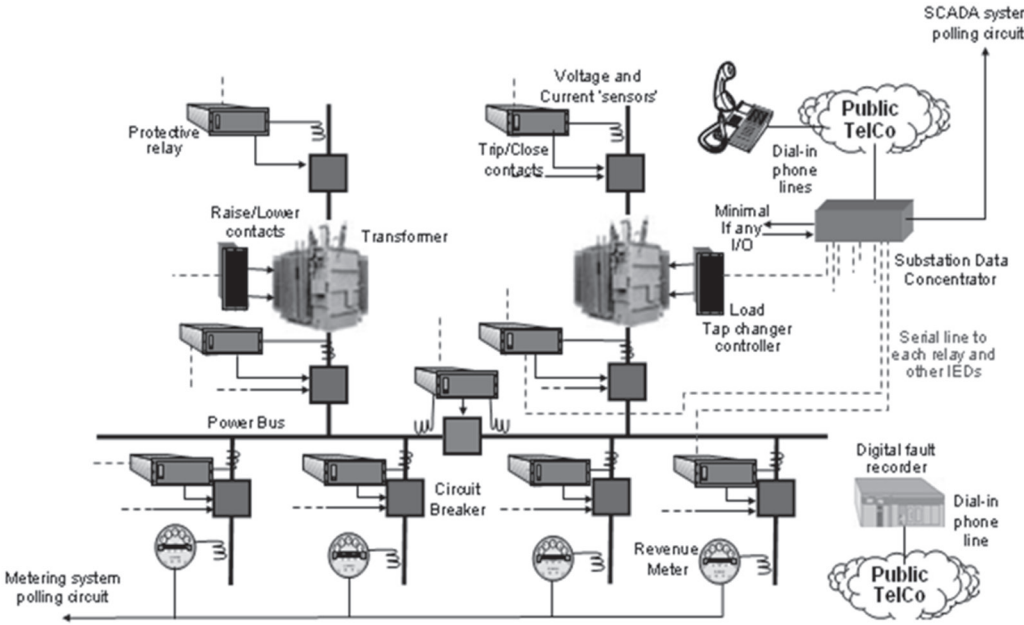


Fig. 12–3. Substation information consolidation (substation automation)

IP to the Substation

In an effort to consolidate communications onto a single circuit (preferably VPN-protected) into the substation, various utilities have migrated onto WAN technologies. Small electric utilities, such as small municipal utilities (*munis*) or rural electric cooperatives (*co-ops* or *RECs*) often have an easier time migrating to networking technologies, as they have a small number of substations and can often string fiber-optic cables along their lines and rights-of-way because of the comparatively short lengths and limited numbers involved.

Many munis are multiple-service utilities (supplying, for example, power and gas or power, gas, and water) and use their fiber-optic bandwidth to offer telephone and Internet services, as well as providing the municipality itself with networking and communications. In these instances, the IP to the substation is over a private network (a fiber-optic one, to boot), and thus it is already somewhat physically secure. The major threat would come from network interconnections between the SCADA system and the other business and IT systems that might have Internet connectivity.

Another threat would be that such a system will have communications hubs at various points around the municipality, possibly in public buildings. Access to the equipment at these hubs must be protected and monitored. These hubs would

need to be considered as belonging within the physical security perimeter of the SCADA system.

Getting physical access to the communications and networking equipment could provide an attacker with communications access to all substations. Of course, if VPN technologies (including authentication via digital certificates) were employed at all substations, mere physical access to a network communications node would not afford the attacker with exploitable communication access. Regardless of whether the WAN connection to the substation is private or leased, the main issue is having IP connectivity to substation equipment (e.g., an SDC) that is designed for IP connectivity and networking (fig. 12–4).

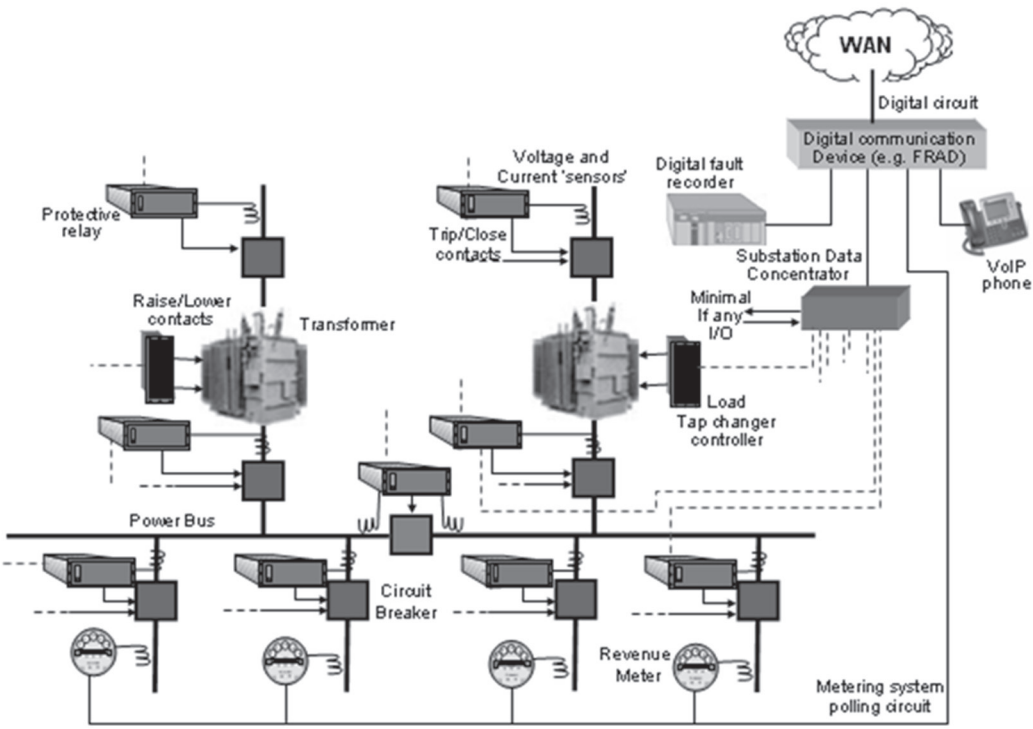


Fig. 12–4. IP networking to the substation

For larger utilities, which cannot reasonably construct and maintain their own communications infrastructure or which need to convert from leased analog telephone lines to leased digital circuits, the challenge is to use public/leased network connectivity in a secure manner. Telecommunications providers can offer a range of wired and wireless digital networking alternatives including (full and fractional) T1/T3 circuits, and even DSL technologies. (Those DSL circuits route your communications onto the Internet, however, which is something of a risk.)

RTUs and SCADA systems capable of supporting IP-based protocols and LAN (Ethernet) interfaces have been available for more than a decade. The commonly used IP networking protocols are primarily the IP version of DNP and Modbus, but there is also a growing implementation base of systems and devices that support UCA2 and ICCP. Extension of IP networking to the substation implies that there is a LAN (invariably Ethernet) in the substation along with IP routing hardware and software. With IP to the substation, a person (or attacker) at a substation could, in theory and unless prohibited via technical countermeasures, connect to the SCADA central computer, corporate servers, and engineering systems. Likewise, a person in their office at headquarters could browse to a substation—and even a specific IED—and have access equal to being present in the substation. When true IP networking is deployed to the substation, there are benefits (e.g., immediate access to fault data and relay settings from the engineer's desktop), as well as dangers (an attacker who gains network access could find his way to the substation IEDs). However, with IP networking, there is the option of employing all available IP cybersecurity technologies: firewalls, VPN, IP_{SEC}, SSL, digital certificates, strong authentication, and encryption.

TASE.2/ICCP Connections

Electric utility SCADA systems often need to interconnect with the SCADA systems of adjacent utilities and with regional coordination and grid management systems (including ISOs and Regional Transmission Organizations [RTOs]). These interconnections are used for data exchanges of real-time data, generation of scheduling data, and even control commands. Many years ago, an effort began to develop a standard protocol for use in establishing these data interchanges. The results of many years of work is ICCP (the Inter-Control Center Protocol), also called TASE.2 (or IEC60870-6 TASE.2). This protocol was initially designed to be transported by an ISO/OSI network, but as that protocol was put to rest in an early grave by TCP/IP, all current implementations of ICCP are IP networking-based (although there may be OSI messages carried by IP, since some vendors took a shortcut in converting their implementation to run over IP networks). At present, ICCP does not incorporate comprehensive security measures (aside from application authentication using association control service element [ACSE] functions, which older versions may not support). Because the ICCP messages are transported by IP, however, other security mechanisms, such as site-to-site VPNs, can be used to protect the communications.

Efforts are under way to expand ICCP and include full security features and functions. Even though the ICCP specification defines eight basic function blocks for ICCP, not every implementation configures all eight. (Some function blocks are quite sophisticated and are used for things like the exchange of scheduling/planning information.) But similar to the use of OPC in DCS systems, the vast

majority hardly implement more than the basic measurement exchange blocks—such as blocks 1 and 2 and possibly data set support blocks 4 and 5. What is to be supported at both ends is defined in the *bilateral agreement* established between the two SCADA systems. Thus, any attack on an ICCP communications link (or through one from a backdoor in the interconnected system) would be constrained to merely abusing the function blocks actually supported (presuming that a firewall at each end would block all but ICCP message traffic). Of course, if remote setpoint control were part of the data exchange, then the consequences could be more significant.

RTUs have become more powerful, and some now support ICCP as an available IP protocol. In this case, the block(s) associated with supervisory control will generally have been included in the ICCP implementation. Thus, an attack on such a connection could make field equipment (circuit breakers, transformer LTCs, etc.) vulnerable, unless some form of additional cybersecurity (such as a VPN with authentication) were incorporated.

UCA2 (IEC61850)

Over the past three decades, another communications architecture and set of standards have been under development, for communications between and among IEDs in a substation and between SCADA (and other) systems and those substation devices. The results of this work are the utility communications architecture (UCA; currently version 2 [UCA2]). UCA2 and ICCP are related: both build on the object-oriented manufacturing messaging service (MMS) protocol and object set (which were originally part of the failed efforts to establish MAP as a factory-floor manufacturing protocol architecture and standard). Some people now refer to ICCP as UCA1.

UCA2 (in a substation) is designed for a high-speed (100 Mbps) switched Ethernet LAN (possibly redundant), to support very high-speed messaging (called GOOSE messages) between protective relays (as a replacement for hard-wired I/O signals). Because of the need for high speed, switched Ethernet hardware support is required for GOOSE support. But many simple substation devices [IEDs] still do not support UCA2, or they support it via an external protocol converter device (that may cost more than the IED itself). These less sophisticated devices still tend to support serial protocols, such as Modbus or DNP3.0. However, market demand for plug-and-play device compatibility is driving the acceptance of UCA2 and its European counterpart, IEC61850.

The UCA2 protocol architecture does not, in and of itself, incorporate any comprehensive cybersecurity features or functions. However, because the protocol requires significant bandwidth (partly owing to the IP underpinnings) and IP networking to the substation, it is not as frequently used to communicate outside the substation. For munis (municipal-owned/operated utilities) and electric co-ops

that create their own fiber-optic WANs, UCA2 can be used for all of the communications between and among their SCADA systems and substations. In that event, physically securing the communications equipment and using IP security mechanisms would be necessary. To move data from a UCA2-compliant substation to a SCADA system, where a high-speed WAN is not available, it would be more common to use a protocol like ICCP, and such a connection (as was discussed previously) can be secured with various IP security mechanisms. It would also be common to see a SCADA system communicate to such a substation through a protocol translator that provided only the necessary information and control points (a subset of all of the available data) through a standard, low-speed, serial protocol, such as DNP3.0.

DNP3.0

DNP3.0 was initially a serial, character-oriented, asynchronous RTU protocol, initially introduced into the public domain by the Harris Corporation and which has been expanded, by the committee that controls its definition, to incorporate a multitude of data types and functions. Many years ago, an IP version was defined (which mainly looks like the same messages being carried as data in IP packets). Currently, DNP3 is one of the most commonly implemented serial protocols used in IEDs by the electric utility market. (Modbus is the other.) Although it is a useful and well-proven protocol, it remains essentially an RTU protocol based on the predefined data and command types it supports.

DNP supports and defines three communication modes: polled, polled report by exception, and unsolicited report by exception. The IP version now incorporates both authentication and encryption functions so it can be used securely over an IP network. Many utilities have converted to DNP3 for their RTU communications, in an effort to use standards and to replace older, less feature-rich legacy protocols. DNP3 works well over slow-speed analog communication circuits. Because it is a well-defined, published standard, DNP3 protocol test sets and protocol analyzers are readily available and can be used to monitor and record ongoing communications and to create and transmit simulated messages to either an RTU or a SCADA system. The same is true for the IP version.

NERC C.I.P. Compliance

After 9/11, in cooperation with various governmental organizations and laboratories (e.g., NIST—the National Institute of Standards and Technology), the North-American Electric Reliability Corporation (NERC) embarked on the development of recommendations and standards to make the SCADA systems and related electrical power grid infrastructure more secure against terrorist threats,

cyber and physical. Its first effort was the Urgent Action Standard 1200 (released in August 2003), which defined 16 areas of concentration, ranging from physical security and electronic security to incident reporting and recovery planning.

In short order, that document was updated and replaced by Standard 1300—Cyber Security (released in September 2004). Standard 1300 consolidated the 16 areas of concentration into just eight, added definitions, and eliminated redundancies. In early 2005, the eight sections of Standard 1300 were individually reissued as CIP-002-1 through CIP-009-1 (Critical Infrastructure Protection). NERC only deals with issues relating to non-nuclear electric utilities and the reliability of the electric power grid, and the standards have been evolving continuously since their initial issuance and now extend to CIP-14-2, which addresses Physical Security requirements. The CIP standards are available on line at: https://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx.

13

Water/Wastewater industry–specific cybersecurity issues

The water and wastewater industries are also major users of SCADA technologies. As the collection, transportation, treatment, storage, and distribution of potable water usually involves a moderately large geographic area, SCADA technologies offer the ideal means for supervising and controlling the set of processes and operations involved. Wastewater (sewage, rainwater, industrial effluent, etc.) collection, transportation, and treatment systems also involve a lot of equipment and sites scattered over a large geographic area (usually the same, or a subset of the geographic area served by the potable water system). Not surprisingly, these geographic areas tend to be municipalities, ranging in size from small towns to large cities.

Both water and wastewater systems involve distributed storage sites interconnected by pipelines with pumping facilities to move the liquids through the pipelines (sewers and mains). They both also utilize processing/treatment facilities located at key points in the network. After that, the similarity ends, but from a SCADA standpoint, the system architectures are pretty much the same. There are obvious differences in the supervisory applications employed, but those don't tend to influence the overall SCADA system architectures.

The supervision and control of water/wastewater processes is not generally an activity that requires high speed and rapid data updating. It has been very common for water/wastewater utilities to use private radio systems for communication with their RTUs and to have polling cycles (the amount of time required for a complete update of field data from all RTUs) of multiple minutes (as compared to multiple seconds in electric utility applications and multiple tens of seconds in the pipeline industry). Large numbers of RTUs (or even all of the RTUs) may share a single, common, low-speed communication channel. Communications between the SCADA system and other systems, such as the control systems running the treatment facilities, are usually via dedicated links (formerly leased telephone lines, but today possibly via the Internet). SCADA systems may or may not be integrated into higher-level business systems and IT applications. Because the service area for a water/wastewater utility is within a municipality, it is usually possible for them to use cellular connectivity. Similarly, it may be possible for them to turn to cable TV providers or the local telephone company to provide Internet access at various sites.

As a municipal utility, the water/wastewater organization often will share a common WAN/LAN with other city departments and organizations. This opens up the possibility of an IP connection that can be an entry point for a cyber assault, particularly as there will be an Internet connection at some point to support Web browsing (possibly a municipality-run Web server/site) and email. The necessary firewall protections (or a DMZ) should be in place on any IP connections to or from a SCADA system and any other network or system.

Communications to RTU sites (and treatment plant control systems) will generally have a primary communications mechanism and a backup mechanism, particularly to critical RTU sites. Today, WiMAX (IEEE 802.16) wireless technology and mesh radio networks with repeaters (based on IEEE 802.11, 15, and 16) are also available technologies for building a private digital network, and they do not require an FCC license and a pre-allocated frequency band. Above-ground storage tanks provide an excellent location for mounting radio repeaters. If the primary communication system fails, the SCADA system can switch over to cellular modems as a backup or, falling back to older technology, poll key sites using analog modems by going through a telephone number dial-out sequence to each such site, one at a time, to obtain updated data values. This might be slow, but it is often sufficient to keep the water/wastewater system operating. The objective is to have an acceptable backup communication scheme that allows the SCADA system to function, even if at a reduced level of performance.

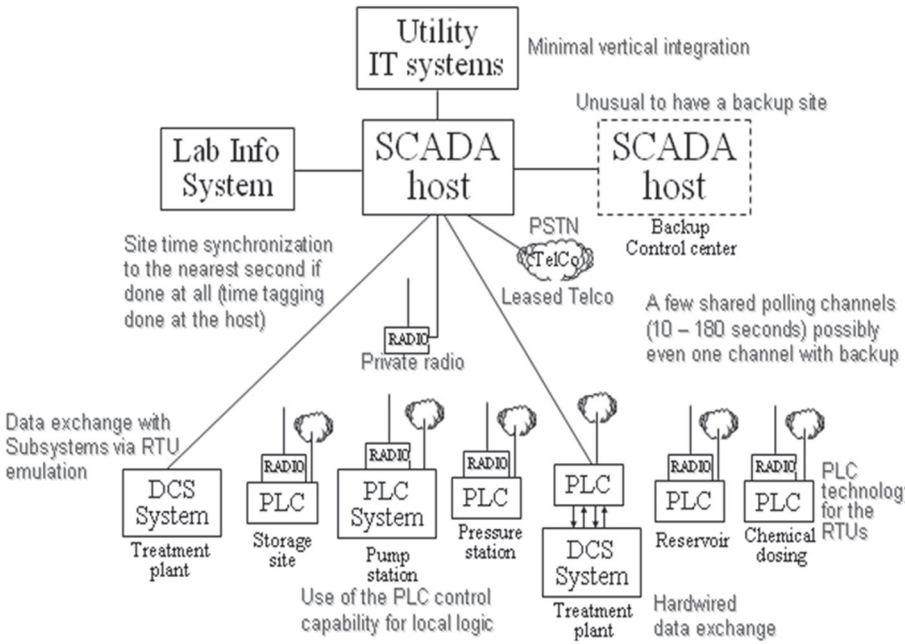


Fig. 13–1. SCADA system block diagram for the water/wastewater industry

Water/wastewater utilities, like electric utilities, were major users of leased analog telephone circuits. We have already discussed the issues related to security (or lack thereof) on analog telephone circuits. Leased telephone circuits might still be used to connect with RTUs or to remotely located master radios, geographically distant from the SCADA facility, usually because of the topology or size of the service area. (The topology might not allow radio communications to all points from a single master radio, so additional master radios can be remotely located, with leased telephone circuits connecting them to the SCADA system's polling channels.) Water/wastewater utilities have used hybrid communication schemes involving both radio and leased telephone lines (not as a backup for each other, but in combination as the primary communication mechanism). Water/wastewater utilities have the same problem with obtaining and maintaining their leased analog telephone lines as do electric utilities, and most have similarly been forced to migrate to digital networking and communications.

Licensed Radio Communications

Because the water/wastewater operations are usually geographically constrained to a metropolitan area, a radio communications system can usually provide the necessary communications coverage. Conventional, single-frequency, half-duplex, licensed radio systems can easily cover an area within a 20–30 mile radius, depending on topology, the transmitter power, and the height and type of the antennas. Often, there are portions of the water system (e.g., wells and reservoirs) that may lie outside the normal service area of a metropolitan telephone company; radio may provide a way for the water utility to establish communications with those sites. Water utilities often have elevated finished water tanks (e.g., "golf ball" tanks) situated along the water main lines for storage and pressure maintenance purposes. These make ideal places to mount radio equipment (and to attract lightning strikes) because they tend to tower above all but the highest structures, and, of course, the water utility owns them. If a repeater-based mesh radio system is used, these tanks often make ideal spots for the geographically distributed repeater stations.

A digital radio system can usually provide a moderate data bandwidth (19.2 kbps in a 25 kHz channel and 9.6 kbps in a 12.5 kHz channel), less than with wireless broadband, but much better than with old analog radio or telephone circuits. All RTUs are equipped with a digital radio modem and antenna. The connection to the radio modem from the RTU can be RS-232, RS-422, or even USB. The range for a digital radio is between 10 and 40 miles, based on topology, and they can be used in either the VHF- or UHF-licensed radio bands. Today of course the alternative is to use unlicensed radios and repeaters that use the ISM frequency bands, but these can be subject to interference from nearby radio sources operating in those same frequency bands.

The equipment used for radio communications is sold by numerous vendors and is totally compatible as long as you know the assigned frequency. Because the frequency is exclusively assigned (licensed) to this application (in this geographic area), it is registered with the FCC and maintained in their records. There are only a limited number of frequency bands and subchannels available for voice and/or data communications (also called *radio spectrum*), and in major metropolitan areas, these are coveted by police and fire departments, taxi companies, and other emergency response organizations and utilities. The upshot of this is that it isn't difficult to find out the frequency (channel) assigned to a water/wastewater utility and to buy commercial radio equipment that will let you monitor RTU polling activities. With such radio equipment and a computer running a serial protocol analyzer application, it is easy to capture and view the message traffic between a SCADA system and its RTUs. Again, we reference the case of Vitek Boden and his wireless compromise of a SCADA system in 2001.

In a basic radio-based system, the central SCADA system normally has an omnidirectional antenna, whereas the individual RTUs usually have directional (Yagi) antennas pointed toward the main antenna of the central SCADA system. (A more complex system might have multiple locations where a radio repeater and antennas are located, in order to reach greater distances or to get around physical obstructions.)

Since radio equipment and laptop PCs are readily portable, a drive around a metropolitan area would eventually lead to the successful capture of complete message traffic. Since most water/wastewater utilities have been migrating to a small number of standardized serial protocols, a knowledgeable eavesdropper would not have too much difficulty identifying the protocol in use. Once that is known, the appropriate protocol test set software and the necessary portable radio equipment would allow the reception and interpretation of polling messages, as well as the generation of syntactically valid messages or responses. There are well-documented cases of this approach being used to issue controls and commands to field equipment by sending false messages to the respective RTUs. Conventional analog radio equipment incorporates no protective or security measures, and the most commonly used RTU protocols don't incorporate any either. Thus, a major cyber vulnerability of water/wastewater industry SCADA systems is the hijacking of RTUs that use conventional, licensed radio communications (fig. 13–2). Using digital signaling versus analog signaling improves message reliability and accuracy, but it does nothing to add cybersecurity. And conventional serial SCADA protocols do not incorporate cybersecurity protections. If wireless mesh networking is used, this kind of radio equipment can incorporate spread spectrum frequency-hopping technology, but as we have discussed in an earlier chapter, that is also not a cybersecurity protective mechanism.

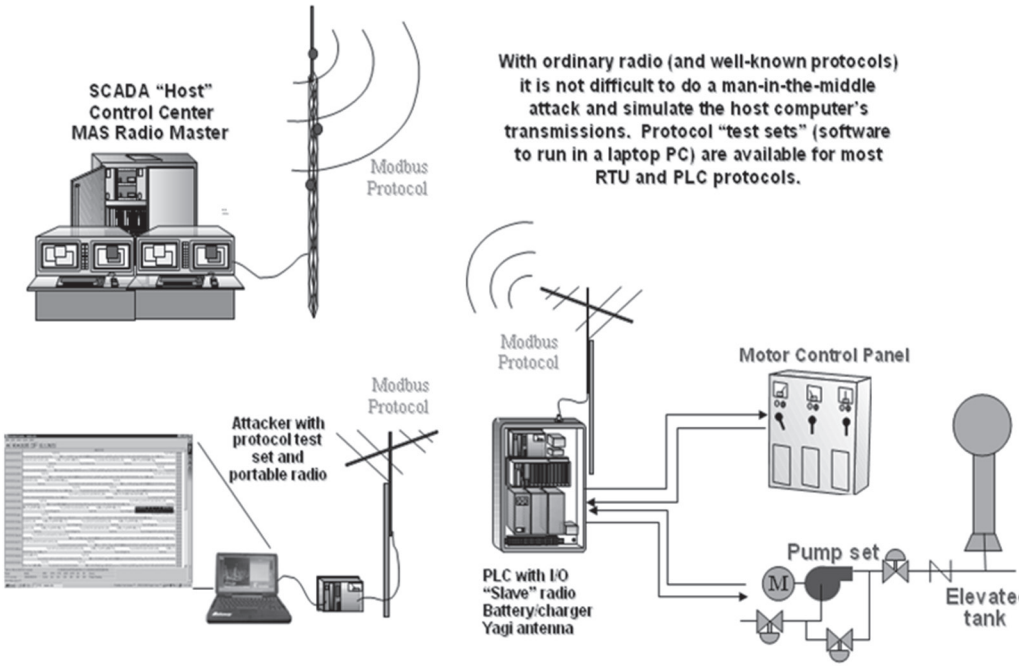


Fig. 13–2. Using portable equipment to hijack a remote site

Nonsecure Protocols

As was mentioned previously, the vast majority of all water/wastewater RTU serial communications—regardless of the underlying communications technology or RTU technology—have employed a well-known serial communications protocol designed for efficiency and compact message size, rather than for communication security. The Modbus protocol is widely adopted by the water/wastewater industry throughout the United States. In part, this is because the Modbus protocol is in the public domain and is simple and easy to implement; also, the water/wastewater industry has adopted PLC technology for use as RTUs, and all major PLC vendors offer the Modbus protocol as either a standard feature or a readily available option. As has been mentioned there are link encryption devices that can be used to secure serial communications. In municipalities where a high-bandwidth IP-based WAN has been created (wired and/or wireless) and made available to the water/wastewater utilities, utilities have used IP-based protocols, including UCA2, DNP3/IP, and Modbus/IP. As was discussed in the chapter on electric utilities, among these protocols, only the IP version of DNP3 currently incorporates comprehensive security features—but since all of these protocols are transported by IP networking, they can all be protected

by employing such standard IP cybersecurity technologies as IPsec and router-based VPN tunnels.

PLC Equipment as RTUs

During the 1970s and 1980s (and even into the early 1990s), a lot of SCADA vendors entered the market, targeting at least one of the three main industry segments; unfortunately, most of these vendors eventually disappeared. This left a large number of installed-base SCADA customers without any support or the ability to expand or modify their systems. At the same time, in industrial automation (and factory automation), PLC manufacturers were introducing and expanding their product lines and slowly adding to their capabilities. Many water/wastewater SCADA system owners discovered that they could successfully apply PLCs to the monitoring, automation, and control problems at their treatment plants (in place of the much more expensive and complicated DCS systems), and also at their pumping/booster stations, and this eventually led to placing the same kind of PLC equipment at field sites where RTUs would traditionally have been installed. To capture the RTU replacement business, PLC vendors and their partners manufacturing product accessories were willing to implement other legacy (serial) protocols. Eventually, the vast majority of water/wastewater utilities adopted as standard one of the major brands of PLC equipment for use as their RTUs and one of the de facto protocol standards (often Modbus serial or IP protocol). If you look at the public-bid procurement specifications issued today by most water/wastewater utilities, you will find technical specifications taken directly from the product literature of their preferred PLC manufacturer.

PLCs began life as simple devices with only contact I/O and relay ladder logic (and no communication capabilities). But over time, PLCs have evolved into flexible, user-configurable, autonomous monitoring and control computers, with a wide range of sequential and regulatory control functions and features. Today, they can support a range of both serial ports and protocols as well as Ethernet ports and IP protocols (fig. 13–3). Since they were initially sold as independent, stand-alone devices, vendors had to develop programming and configuration utilities that provided for the downloading of control logic and configuration information, either through a direct physical connection to the PLC or via the communication link to the PLC. When PLCs were originally introduced, they were targeted for factory-floor automation, so it was no big deal to physically reach a given PLC to establish a communication connection to its programming port, for downloading logic and configuration changes. As PLCs were scattered to remote sites, with only the polling channel available for communications, it became necessary to provide protocol messages/codes that supported logic and configuration downloading into the PLC via the same communication channel used for the polling, to avoid the logistical nightmare of going to numerous, widespread PLC sites to make changes. At a remote site, there may be a single PLC acting in the role as an RTU. At a

treatment plant or pumping/booster station, there may be a small control system built out of multiple PLCs connected by a LAN (probably Ethernet today). In this case, one of the PLCs may act as a data concentrator and collect status and data from other PLCs so that it can be made available to the SCADA system via a single communication interface. If the municipality has run IP networking to the site, then the PLC LAN may just be bridged to the IP network via a router.

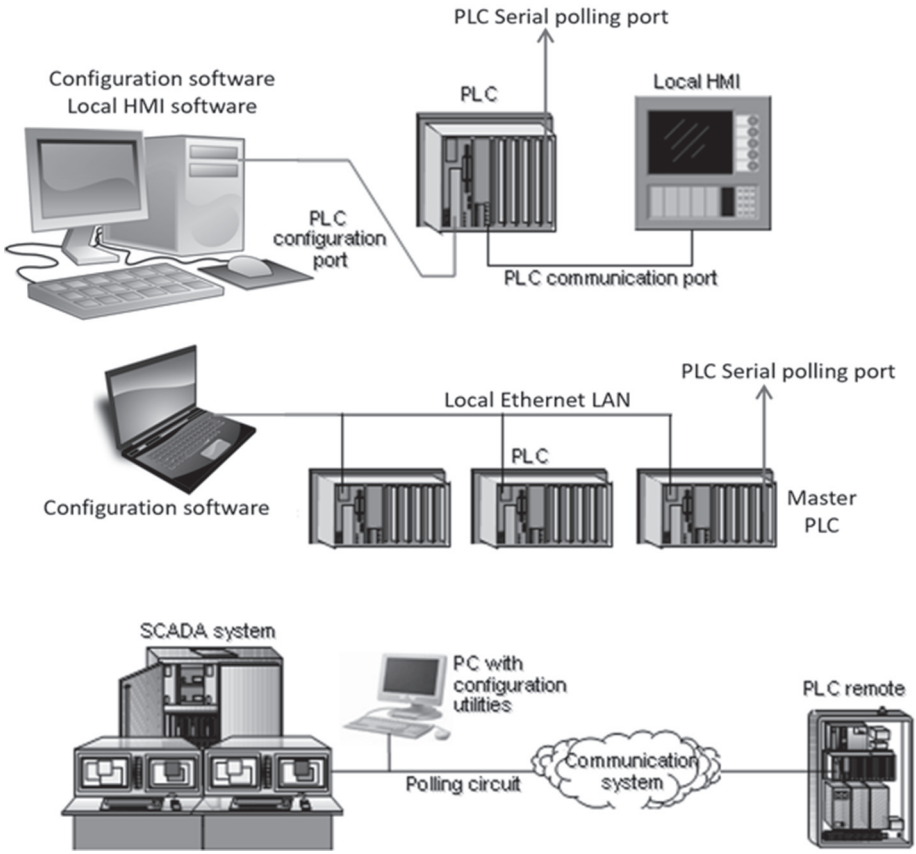


Fig. 13–3. Evolution of PLC programming and configuration downloading

Supervisory and Local Control Applications

The electric utility market makes great use of industry-specific application software, but little of that software actually initiates closed-loop (automatic) control. Sending out generator unit raise/lower commands or generating setpoints is the major type of closed-loop supervisory control in the electric utility industry. There is also little local autonomous regulatory control, other than load tap changers,

voltage regulators, and a capacitor bank here and there (and sectionalizing in distribution systems). Further, the control logic for these types of devices is usually embedded and not remotely alterable.

By contrast, the water/wastewater industry frequently uses supervisory and local control applications to make decisions about starting and stopping pumps, opening and closing valves, and performing other closed-loop autonomous operations such as level control and flow and pressure regulation. Many operations in the water/wastewater industry are based on time of day (or type of day: weekend, holiday) or are triggered by alarms/events (e.g., sudden rainwater inflow) or are initiated by operator command. The actual control logic may be a program (or supervisory control script) that runs in the SCADA central computer, or it may be control logic residing in the PLC(s). If an attacker can cause the right signals to be sent to a PLC, the local control logic can be triggered; if an attacker can cause a simulated alarm/event to be sent to the SCADA system, supervisory control logic can be triggered; and if an attacker can get programming access to the SCADA system (as an insider might have), then a malicious supervisory control program could be added to the system and initiated by events or conditions defined by the attacker (a.k.a. a time bomb application). Physical security measures at the field sites and the SCADA control center, along with operational controls (e.g., separation of duties: application coding and testing) and applicable cybersecurity measures can minimize the likelihood of a successful cyberattack. But if the communication to the field sites is serial and cannot be protected via encryption, then this is the major exploitable attack pathway, and there are examples of this occurring, particularly if the communications are radio-based. There are point-to-point (and multi-point) serial encrypting devices on the market today (and have been for many years). These devices can perform AES encryption using keys of 128 or more bits and can work at serial transmission speeds of 1200 bps up to 56 Kbps and can operate from either AC or DC power sources and accept RS-232, RS-422, or RS-485 inputs. These devices would need to be installed at each field site on the polling circuit and on each of those same polling circuits at the SCADA master. Installation coordination would be needed, as once installed at one end of the circuit, all communications would be impossible until a matching unit was installed at the other end of the circuit. And the issue is far more complex for multi-dropped circuits. Manufacturers such as Data Comm for Business (www.dcbnet.com/datasheet/sseds.html) and SEL (https://cdn.selinc.com/assets/Literature/) produce these "bump in the wire" link-encrypting devices for SCADA applications.

Local automatic control has been a particular strength of PLCs. Often, a local PLC will have preprogrammed emergency shutdown and startup logic that can be triggered by conditions it is monitoring or by *flags* set by the operators. There is no authentication between the PLC and the operator; if a syntactically valid message arrives, the logic will be triggered, regardless of where the message actually came from. (If the communications are Ethernet-TCP/IP-based, then industrial firewalls CAN be used to provide some protections, but IP addresses can be spoofed.) Some

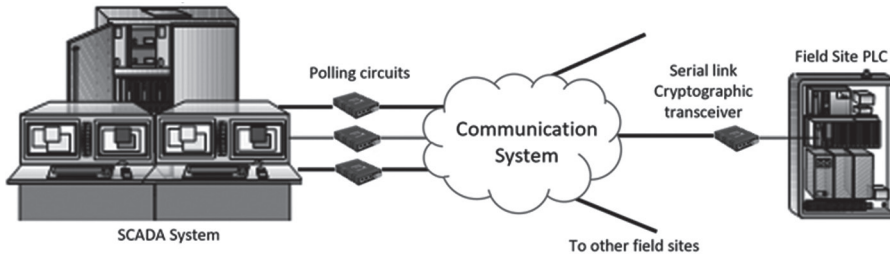


Fig. 13-4. Using serial link cryptographic transceivers

logic designers, recognizing the possibility of human error, will include a requirement for a second, confirmation message (within a given time window) in order for the logic to be triggered. (For a discussion of select-check-operate message sequences from early dumb RTU technology, see section 1.) A patient attacker can eavesdrop on unsecured RTU/PLC communications and eventually record such sequences. Simple serial protocols, such as Modbus, are highly vulnerable to a replay attack whereby the previously recorded message (or message sequence) is merely regenerated on the communication circuit, a capability standard in most protocol analyzers or protocol test sets.

It is possible to improve on communications and control security by using additional application logic (which would require additional logic at the host level and within each PLC). As an overly simplistic example, if the PLCs and the host are kept within a reasonable time synchronization in some manner, such as having the host send out time broadcasts on a periodic basis, then it could be possible to require current date/time to accompany control messages (as a separate pre-command message). The PLC could compare its time to the time in the message, and if they differed too much (as in a replayed message recorded earlier), it would discard and ignore the message. You could also require a set of flag bits to be set in the proper value and order to enable a following control command. Obviously, this kind of logic complicates the SCADA system and the PLC logic. And any scheme that involves a series of messages can, in theory, be recorded and analyzed to figure out what you are doing. But any scheme that makes it more difficult to attack the field devices and issue malicious control actions is a good thing to consider.

Efforts to develop secure, low-speed, serial communications protocols were under way several years ago (the AGA even proposed a lightweight encryption algorithm), with the goal of adding encryption to many popular serial protocols. But the migration to broadband and IP to the field, and conversion of standard serial protocols to IP-based versions, short-circuited that effort. If communications are converted to digital networks that support IP protocols, even wireless ones, then security technologies are readily available already.

The results of an attacker invading a water/wastewater SCADA system might not be as newsworthy as a major electrical power outage or a major pipeline leak.

Nevertheless, damaging a water-delivery system, cutting off water to a major metropolitan area, and causing a major sewage spill are all possible outcomes of an attack on a water/wastewater SCADA system. Thus, the potential exists to alarm—and even harm or kill—a large number of people.

Municipal LANs and WANs

Water/wastewater utilities, as was already mentioned, mostly exist for the support of residents and businesses of the municipalities in which they are based, as well as for the municipal government and its facilities. Therefore, they frequently share the communications infrastructure that supports the municipality, especially if that infrastructure was funded and constructed by the municipality. It is not uncommon to see municipalities that have invested in high-speed, IP-based municipal area networks (MANs) using fiber optics and networking technologies such as FDDI, SONET, or ATM. The bandwidth of such a WAN may be sufficient to offer telephone, networking, Internet, and even cable TV services. The municipality can often fund the cost of creating and maintaining a MAN by the sale (lease) of these various services, leaving plenty of bandwidth available for their own voice and data communications needs. A municipal water/wastewater utility is equally likely (as is the municipal electric utility) to utilize communication services over such a WAN, even if that addresses only part of their overall communications requirements. One cybersecurity-related problem with such an arrangement is that many times the MAN will have communication nodes in public buildings or commercial buildings shared with public companies (see fig. 13–5). As was mentioned in the section on municipal electric utilities, this places physical access to the network equipment into locations that need to be protected and included within the physical security boundary of the SCADA system.

Another problem is that the MAN will often interconnect with conventional IT systems for both the municipality and major utility customers/suppliers, and somewhere out on the MAN an attached IT system will make a connection to the Internet. The security issues—and the strategies to meet those challenges—are essentially the same as those of a municipal electric utility. (For details, see Chapter 12.) One aspect of a MAN that wasn't included in the discussion of electric utilities is that there may very well be sub-multiplexed voice-grade channels (e.g., analog PLC communications traffic) passing through the MAN and interfaced to public, leased telephone lines to reach outlying sites. These interfaces are a point of vulnerability, unless the serial communications are encrypted (at the host and the PLC ends), and also physically secured to prevent access by unauthorized personnel. When a portion of the bandwidth of a digital network is sub-multiplexed to create voice-grade channels (for actual telephone service or for data communications), those channels must be treated no differently, from a security standpoint, than telephone lines leased from the local telephone company.

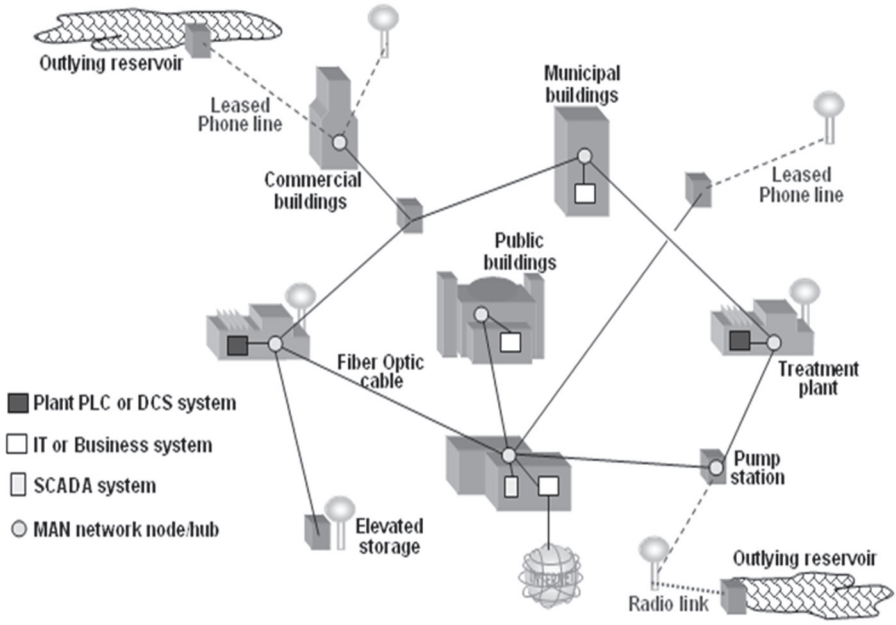


Fig. 13–5. Example MAN shared by a municipal utility SCADA system

One way to enhance the SCADA system security in a shared MAN arrangement is to use VPN technology among the SCADA system and water/wastewater system components (primary and backup SCADA control center, treatment plants, and pumping/lift stations) so that they are essentially on a private MAN of their own. You could attempt to use VLAN technology to isolate these portions of the MAN, but as we discussed in a prior chapter, VLAN technology can be compromised.

Control Interfaces to Plant Control Systems

Another difference between water/wastewater and electric utilities is that treatment plants in a water/wastewater SCADA system may be tightly coupled and interfaced to the SCADA system, with significant remote control and adjustment capabilities, because the status and operational level of a given treatment plant can have a significant impact on the overall operation of the water/wastewater utility. (An off-line, or load-restricted, treatment plant can result in the inability to meet water demands or the need to allow sewage to accumulate—or, unfortunately, to be discharged untreated.) The level of automation in a modern water/wastewater treatment plant can be quite high, nearly eliminating the need for on-site supervisory or operational personnel—except during peak-load hours and in the event of plant problems. Therefore, it is common for the interface between the SCADA system and the individual treatment plant control systems to include a large volume

of real-time operational data from the plant and a good number of supervisory controls and adjustable parameter settings going back to the plant control system from the SCADA system. This is so that SCADA personnel can monitor and (to a degree) remotely adjust/control the treatment plant, if necessary. Depending on the computer technology of the water/wastewater treatment plant control system, there may be remote viewing and control capability provided in the form of remote operator console support (possibly Web browser/server-based). This type of temporary remote access, if supported, should be safeguarded using the various IP security technologies already discussed, such as setting up a VPN tunnel. If such security measures are not employed, then an attacker who gains access to the communications channel has a reasonably good chance of gaining full operational access to the control system. If a remote operator has a Web browsing session to the control system and has already been authenticated, then an attacker who hijacks the session has the same control authority (and sees the exact same displays) as the operator.

14

Pipeline industry–specific cybersecurity issues

The pipeline industry can be decomposed, at the highest level, into liquid and gas pipelines, and those two broad categories can be further decomposed. Gas pipelines carry natural gas (methane) or other gases, such as carbon dioxide (CO_2). Natural gas pipelines are often quite long and can run halfway across the United States, whereas a CO_2 pipeline might just run a few miles between the plant where it is produced and another plant that needs it as feedstock. Liquid pipelines can carry crude oil to a refinery from a production field or a tanker unloading terminal. Liquid pipelines may carry a single refined product, multiple refined products (in batches), or liquids produced from the processing of natural gas (NGL).

There are obvious differences between the various types of pipelines, particularly in the operation of the lines and the control strategies and supervisory applications (one major difference being the high compressibility of gases versus the near incompressibility of most liquids). However, from a generalized communications and SCADA viewpoint, these different types of pipelines tend to be quite similar: they will have measurement and control points distributed along the pipeline, as well as locations where the transported medium is injected into the pipeline and where it is extracted from the pipeline. Both liquid and gas pipelines also need occasional booster/pump/compressor stations to maintain the required flow rates and pressures along the line. A SCADA system for a pipeline will usually have RTUs located along the line to make measurements and provide remote control of pumps, valves, and other equipment, such as for cathodic protection and meter runs. Often, some number of these RTUs will be situated at measurement points with no power source and thus will have to be solar- (or alternate method) powered and designed for very-low-power operation, including their radio equipment.

One particular variation in production field SCADA systems is the offshore (oil/gas) production field, where the RTUs are located on platforms out in the ocean and thus have to survive in this environment. Communications to RTUs located offshore can be disrupted by storms, fog, and hurricanes; thus, the RTUs need to be able to operate independently of the SCADA host for long periods of time. The RTUs used in pipeline applications tend to incorporate more (and

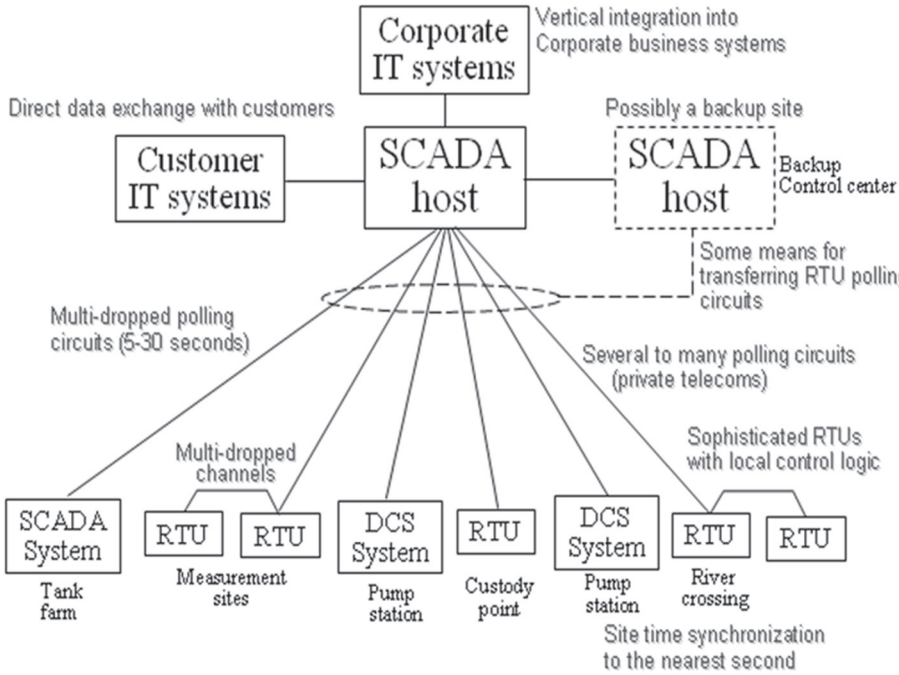


Fig. 14–1. Generalized architecture of a pipeline SCADA system

higher-complexity) calculations and regulatory and sequential control logic, as compared to those used in water/wastewater applications (or in electric utility applications, which generally have few such features).

Pipeline SCADA systems also normally interface with the control systems used to automate the pump/booster/compressor stations and may even provide remote control (or at least adjustment) of such systems (fig. 14–1). For liquid lines, there will also be storage areas (tank farms) at the injection and extraction points. Gas lines may move gas to or from production and storage well fields. Optimization of pipeline operations usually requires that the entire pipeline be modeled; this can be achieved only at a supervisory level, at which continuously updated real-time data is available for the entire pipeline. The output of pipeline models may initiate either the direct remote operation of pipeline equipment (e.g., to start/stop pumps and open/close valves) or the adjustment of operational set points within the booster/pump/compressor station control systems. Because energy trading and delivery schedules—an integral aspect of the business enterprise—rely on current operational information, pipeline SCADA systems are often tightly interfaced with other business systems across the corporate WAN. (The term *vertical integration* is often used for the business segments, but applies to data integration as well). Although pipelines can be used to transport many types of gases and liquids and can come in many lengths, the focus here will be on

the long-haul pipelines used for delivery of hydrocarbon fuels (gas and liquids), since disruption (or disabling) of one of these pipelines would have the greatest impact on the United States.

Radio Communications

By their nature, hydrocarbon pipelines tend to run long distances through rural (or sparsely populated) areas—from the source of hydrocarbon fuels (the Gulf Coast) to places where such fuels are needed (e.g., the East Coast and the West Coast). Because they tend to be located away from public infrastructure, and may cross through the service areas of multiple utilities and telecommunication providers, pipelines have almost always incorporated private communication systems. Until the introduction of fiber-optic cable in the 1980s, communications were radio-based, utilizing three types of radio technology:

- Analog point-to-multipoint radio
- Microwave radio point-to-point links
- Satellite uplinks and ground stations

As mentioned previously, pipeline companies typically borrowed their communications technology from the telephone company. Thus, a pipeline built in the 1970s would probably have included a series of microwave relay towers located along the pipeline right-of-way and used to provide voice-grade telephone communication circuits (for voice and data) along the pipeline and back to the SCADA system (fig. 14–2).

Starting in the late 1980s (and continuing thereafter), microwave towers were replaced by a fiber-optic cable that could be buried in the same trench as the pipeline. (Here I am talking about new construction. Pipeline operators didn't go out and tear down the existing microwave infrastructure. Hybrid systems, where expansions were made with fiber optics, can also be found.) Although the communications used fiber-optic technologies, they were, however, undoubtedly still based on voice-grade telephone circuits generated by de-multiplexing the bandwidth of the fiber-optic cable (in the same way that the microwave signal would have been de-multiplexed).

Today, that very same fiber-optic cable can be used to carry IP networking (including VoIP telephone service) by replacing the telephone multiplexer equipment with Ethernet switches. The placement of microwave relay towers (or fiber-optic repeaters) is dependent on a source of power. Because this equipment needs to be protected from the elements and tampering, its physical positioning cannot always be adjacent to every point of measurement and control. (Microwave towers also need line-of-sight relationships with up- and downstream towers—something that isn't an issue with fiber-optic cable. But when you get to a river crossing, microwave regains the advantage.)

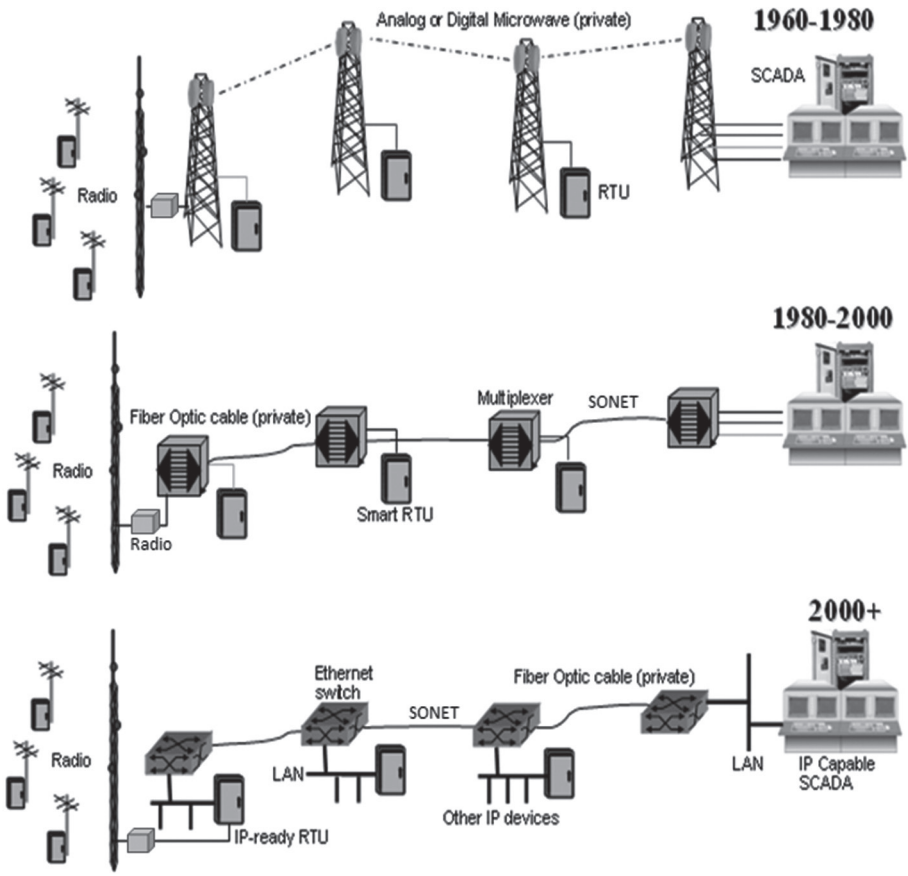


Fig. 14–2. Evolution of pipeline communications technologies

With pipelines, it has been typical to combine conventional radio communications with a microwave/fiber-optic backbone, to extend communications to geographically adjacent areas. The susceptibility of radio communications to tampering (hijacking) has already been discussed; unlike in the water/wastewater industry (where radio is often the primary communications mechanism), along a pipeline, radio communication *spurs* are generally used only to bridge lower-priority measurement sources onto the (microwave/fiber-optic) communications backbone. Unlike conventional broadcast radio, satellite and microwave communications tend to be more physically secure from attack; to tamper with or disrupt communications, you have to have physical access to the equipment or get in line with the highly directional/focused radio signal. (Still, this is a possibility, since many pipeline facilities are located in the middle of nowhere and are not under constant supervision.)

Along a pipeline, the facilities provide physical security in the form of fences, walls, and buildings that house the communications equipment. (Also, much of

the pipeline itself is buried, which gives added physical protection, especially when fiber-optic cables are used.) In theory, you could erect a temporary tower between two microwave repeaters and intercept or disrupt the communications, but this would require significant and rather obvious effort. Disruption of communications would probably be unsuccessful (unless initiated on a widespread basis) in that most microwave systems are designed in a looped configuration that permits communications to continue (by reversing direction around the loop) if any repeater station goes down or if local weather conditions disrupt any section of the backbone. With the advent of fiber optics to replace microwave towers, it is theoretically possible for someone to excavate along a pipeline and reach the fiber cable. Once this were accomplished, the cable could be cut and an additional (de-)multiplexer (or Ethernet switch) could be inserted to gain access to the communications. However, this would be a technically daunting task requiring a lot of insider information to be successful. It would be much easier to merely breach the excavated pipeline with explosives and ignite the gas/liquids therein that then become thereout.

Smart RTUs

The pipeline industry has made extensive use of microprocessor-based RTUs since their introductions in the early 1980s. SCADA vendors who serve the pipeline industry have incorporated many pipeline-specific measurement and control functions in their RTU products, along with general sequential and regulatory control functions. As with the water/wastewater industry, locally and remotely (re)programmable RTUs are used and often contain downloaded calculations and control logic, enabling them to perform autonomous control functions. It is not uncommon for RTUs to handle the opening and closing of valves, the regulation of pressure and flow, and the operation of pumps and compressors. Volumetric metering of gas/liquids for billing and leak-detection purposes is also a common function in pipeline RTUs.

If the control logic of an RTU were replaced with malicious code, it would be possible for an RTU to cause physical damage to the pipeline equipment or to cause intentional product leakage. With a product pipeline, tampering with RTU logic could cause the wrong products to be routed to designated storage tanks, thus rendering the products unusable without reprocessing. All of these scenarios would require communications (or physical) access to the RTUs (or the SCADA host computer), as well as application knowledge to which only an insider would have access.

The pipeline industry (for obvious reasons), unlike the water/wastewater industry, has been more inclined to use separate, hardwired safety equipment to prevent malfunctions of the normal controls (e.g., the RTUs) from initiating a catastrophe. Much of the control logic running in the RTUs (or separate safety systems) is designed to make a determination of safety, based on local measurements, and to opt for the

safest strategy. Thus, totally disabling a pipeline SCADA system might not have as much of an impact as taking control of such a system and using it maliciously. Taking control of an RTU and trying to use it to cause damage might cause external hardwired safety logic to be triggered in order to block or limit the actions of the RTU.

RTU Program Logic

As with water/wastewater SCADA systems, the need to change program logic in a number of remotely located RTUs led the pipeline industry to adopt centralized programming and configuration of pipeline RTUs as early as the late 1970s. To stay competitive, most SCADA vendors serving that market had to roll out smart and remotely configurable RTUs and had to extend their RTU protocols to support these capabilities. Since this technology emerged prior to the widespread adoption of PLCs (and PCs), the programming tools tended to be developed as integral components of the SCADA host system software, not as independent utilities that could be purchased separately. This also meant that there was little or no compatibility between vendors, which eventually led the industry to push for cross-platform standards. Standardization in the pipeline industry has not reached the same levels as it has in the water/wastewater industry, but it is headed in that direction.

On the positive side, security-wise, to use the RTU programming and configuration tools, one would normally have to have access to, familiarity with, and a valid user ID/password on the SCADA system. Today, more and more pipeline SCADA systems are based on commercial SCADA packages, much like those used in the water/wastewater industry, and are using both RTUs and PLCs as remote units. PLCs can now be programmed to perform functions that were once exclusive to industry-targeted RTUs (AGA gas volumetric calculations, tank volume strapping tables with specific gravity and temperature corrections, etc.). This means that the pipeline SCADA industry now is encountering the same problems with securing RTU configuration and programming as did the water/wastewater industry. Programming tools are commercially available to anyone, and RTUs (or PLCs) have no authentication mechanisms to detect and prevent unauthorized programming and configuration changes.

Supervisory Control Applications

Because pipelines are complex and need to be modeled for a variety of reasons, it is commonplace to use supervisory applications, running on the SCADA system, for autonomous, automatic control of the pipeline. Depending on the pipeline operator's operational philosophy, the supervisory control applications may or may not request permission from a human operator prior to sending control messages to RTUs and local control systems. Importantly, these applications seek

operator authorization only because they have been specifically programmed to do so. In most SCADA systems (except those in which device tagging is supported), there is nothing in the system design that prevents a supervisory program from sending a control command to an RTU. It is usually up to the application developers to add specific program logic to request operator authorization (possibly with a timeout that allows program continuation if no human responds to the request for authorization). This means that if a malicious supervisory application could be introduced into the SCADA system, such a program could send commands out to the field devices. The risks posed by malicious supervisory applications and the need for insider access to create and install such applications are the same as have already been discussed in Chapter 12, in regard to electric utilities.

IP along the Pipeline

In the communications architecture evolution shown previously (see fig. 14–2), the most recent step is the conversion of the high-bandwidth backbone into a WAN, by replacing the multiplexing equipment with network components such as switches and routers. This step has made possible the use of the WAN for a variety of applications. With IP-enabled SCADA systems and RTUs, the WAN provides real-time data and command exchanges for the basic SCADA functions.

If the local control systems in the pump/booster/compressor stations are likewise IP-enabled, the WAN offers remote access to these systems for control, maintenance, and support functions. Personnel working at or dispatched to the various sites along the pipeline might even have been allowed email and Web browser access to the corporate servers (because at the time no one was concerned about cyber security). Various corporate business and asset/maintenance management systems may have direct communications with corresponding subsystems along the pipeline. VoIP and *Webcam* technologies can be integrated into the WAN for voice communications and for visual supervision and monitoring of the various pipeline facilities.

In other words, as with the other industries that employ SCADA systems, numerous technical and operational advantages are gained when the underlying communications are based on IP networking. Further, just like those other industries, deploying IP networking to the field without adequate technical safeguards offers the opportunity for hackers and other threat agents to gain access to those IP-enabled systems and devices. Unlike the water/wastewater and municipal electric utilities—where networking equipment could be located in readily accessible public buildings—with a pipeline, the communications equipment would normally be housed in (reasonably) secured facilities (along the pipeline) and would not be readily accessible to the public. Nevertheless, if an attacker were to gain physical access to a remote pipeline facility that was part of the WAN, they could potentially access any other WAN-connected system or equipment, if adequate cybersecurity

protective measures were not implemented. Physical security is never a substitute for appropriate cybersecurity and electronic security measures; it is just part of a comprehensive *defense-in-depth* security strategy.

Web Browsing and Email Integration

A pipeline SCADA system built on a private communications network along the pipeline right-of-way still requires a control room somewhere, staffed with operational and supervisory personnel. It is quite common that this control room be located in the headquarters of the operating company, since as long as communications are available, there is no real restriction on its location. (This is much the same as in the electric power industry.) The people who operate and supervise the pipeline SCADA system need to communicate with other corporate personnel and provide data to related corporate systems (e.g., asset management systems, accounting systems, billing systems, and even maintenance management systems).

It used to be that when a SCADA system was implemented, it was a standalone system that wasn't interconnected with a company LAN/WAN. For many years, concern over the security of corporate networks—and the Internet—kept SCADA systems physically isolated. It remains common to see separate PCs in a control room connected to the corporate network, for email and Web access by the operational personnel. This was easy to justify when SCADA systems were built from computer equipment that markedly differed from the average desktop PC, but today there is little difference (other than in name) between a person's desktop PC and the operator console attached to the SCADA system or one of its servers. It has become more usual to see the SCADA system's LAN bridged onto the corporate network via some form of firewall that permits bidirectional email and Web browsing directly from the operator's console. The only intended purposes of this integration might be to enable internal email (within the corporation) and to provide Web access to corporate servers. However, somewhere on the corporate network, there is a connection to the Internet, because others in the corporation do require World Wide Web and email access. For a long time, no one viewed this *creeping integration* as particularly dangerous, and it was convenient for operational personnel to have these facilities. Today, such functional interconnectivity poses a security risk to the SCADA system because of malware, which can be attached to or embedded as part of an email message or implanted while Web browsing. As already discussed, the danger of creating an attack pathway for threat agents is always present when a SCADA system is connected to other systems and networks that are connected to the Internet. Most SCADA-knowledgeable security experts would agree that functions such as email and Web browsing, if required, ought to be relegated to separate PCs on a separate network, which is not connected to the SCADA system's network (and the dangers of dual-homed PCs, even if via integral wireless networking, have already been presented).

15

The cyberthreat to scada systems

During the last two to three decades, supervisory control systems in various countries around the world, and in various industrial applications, have been subjected to cyberattacks—or, in some cases, a cyberattack was suspected but could not be proven. Even SCADA systems in the U.S. have been subjected to cyberattacks, with varying levels of success; fortunately, to date, nothing serious has happened as a result. One of the problems cybersecurity professionals and various governmental agencies have is getting organizations to provide information when they suspect/detect a cyberattack. Some SCADA organizations don't attempt to make a root cause determination when things aren't working properly; they just reload, reboot, and get the system operating again ASAP. So, if the problem they were experiencing was due to a cyberattack, they won't know and won't be able to perform any after-the-fact forensic analysis. Other organizations worry that they will get a bad public image (and possibly suffer a financial impact) if people find out that that they were compromised via cyber means. Industrial control systems continue to be soft targets for adversaries. According to CyberX and their 2019 Global ICS & IIoT Risk Report:

- 40% of industrial sites have at least one direct connection to the public Internet.
- 53% of sites have obsolete Windows systems such as Windows XP, ME, or 2000, and many do not keep up with patching.
- 69% of sites have plain-text passwords traversing their ICS networks (insecure protocols).
- 57% of sites are running anti-virus protections that are infrequently (if ever) updated.
- 16% of sites have at least one Wireless Access Point.
- 84% of industrial sites have at least one remotely accessible device/computer.

Of course, these statistics are a combination of both sites with DCS and SCADA systems, but the general findings are relevant to both types of IACS and indicate that too many SCADA system owners don't focus on ensuring that their cybersecurity efforts are effective and well-maintained. Too many organizations treat cybersecurity as "one and done," rather than a continuing and evolving effort that has no foreseeable end.

We have discussed that the threats to SCADA systems can come from amateur hackers (script kiddies), professional hackers, nation-state-sponsored hacking organizations, criminal hacking groups, or just from malware in the wild. The seriousness of a cyber incident will depend on what was done, how, and by whom. There are at least seven (7) general categories of cyber incidents:

1. Incidents that turn out not to be cyber-related (incorrectly categorized)
2. Cyber incidents that are non-malicious, usually caused by human error
3. Automated scanning, which happens all the time, often by Web-crawling apps
4. Amateur attacks where the exploits are inappropriate and not stealthy
5. Directed attacks that fail, but are clearly not random and require some skill
6. Thwarted attacks that are at least partially successful in penetrating defenses
7. Successful attacks that partially or totally achieve their objectives

The first few of these categories are commonplace and pose little risk. But the last three categories are clearly bad and the hope would be that your protective measures prevent them from ever happening (or that your incident response and recovery procedures are effective and get you back up and running before anything really bad happens).

- **Directed attacks—**Attacks that are specifically aimed at the facility, show at least a moderate level of skill on the part of the attacker and/or which show a knowledge of some site-specific information which makes it clear that the attack is not merely random or 'fortuitous' (for the attacker). This type of attack indicates that some threat actor is specifically attempting to penetrate your defenses. If such an attack is detected you will want to take swift action to block the attack pathway that was being used in the attack.
- **Thwarted attacks—**Attacks which manage at least a partial penetration of the facilities defenses (which indicates a sufficient skill level on the part of the attacker) but which fail to achieve their intended purpose(s) such as establishing a beach-head, implanting tools, exfiltrating information, etc. This type of attack clearly indicates that some threat actor is specifically attempting to penetrate your defenses. If such an attack is detected and thwarted you will want to take swift action to block the attack pathway that was being used, and any exploitable vulnerabilities used in the attack, are addressed and eliminated.
- **Successful attacks**—Attacks that were partially or fully successful and only afterwards (immediately after, or longer, based on impacts) is the resulting compromise discovered, through various means, and the impacted systems restored and reconstituted. If you are successfully

cyber attacked then clearly you have viable attack pathways and exploitable vulnerabilities that were either not identified or not give adequate protections. Recovery from such an attack will test your incident response capabilities and might lead you to improving them.

Part of your cybersecurity program should include some process or procedure that is used to assess and classify identified cyber incidents. A well-designed and implemented SIEM can be invaluable in sorting out the seriousness of a cyber incident. You don't want to expend much effort in examining cyber incidents that fall into the first three (or four) categories, but you need to pay serious attention to any that fall into the final three categories (fig. 15–1) and take the appropriate actions to protect your SCADA system, determine how the attacker got through your defenses, and take steps to eliminate the exploited pathways and vulnerabilities.

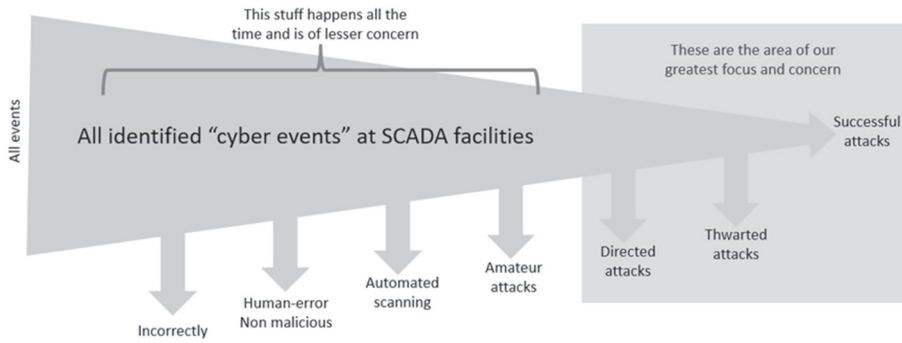


Fig. 15–1. Cyber event categorizations

The British Columbia Institute of Technology (BCIT) started compiling a database of IACS cyber incidents back in the late 1990s and maintained it until 2015. This database contains a wide range of cyber incidents spanning many countries and numerous industrial segments. The DHS in the U.S. is also maintaining a database of cyber incidents, although their database covers all business sectors from banking to manufacturing. The following are a few of the documented cyberattacks on SCADA systems.

Identified incidents

As we have mentioned, cyberattacks may be launched for a variety of reasons. In 2015, the SCADA system running the power grid in Ukraine was attacked by Russian hackers and caused a blackout that affected 225,000. Then again, in 2016, another cyberattack was launched by Russian hackers, and this disrupted power in the capital city of Kiev for about an hour in the middle of the night on December 17 (the middle of the winter). Clearly, this attack was related to the ongoing conflict between these two countries.

Cyberattacks have been launched numerous times on the power grid in Israel going back to 2013 when an anti-Israeli hacker group attempted to penetrate numerous systems. In 2016, a large cyberattack targeted the national power grid SCADA system and caused temporary power disruptions.

The operators of the Alyeska crude oil pipeline in Alaska have recorded numerous attempted cyberattacks on their SCADA system and the DCS systems at the various pumping stations along the pipeline. To date, the attacks have been thwarted, but they continue to be attempted every so often. A variety of groups from nation-states to criminal groups have been associated with these attempted attacks.

Earlier in the book, we mentioned the 2000 hacking of the Maroochy Shire sewage control system, using stolen radio and laptop PC equipment. Vitek Boden (a former employee of the SCADA system vendor) accessed computers controlling the Maroochy Shire Council's sewerage system, altering electronic data in respect of particular sewerage pumping stations and causing malfunctions in their operations.

In 2004, a security audit of a potable water SCADA system discovered that a Trojan backdoor was installed on a human machine interface (HMI) computer. This Trojan contained a keylogger and reverse tunnel to an outside Web site. Because the HMI had Internet access, it was presumed that the HMI was infected by an operator browsing external hot-mail Web sites, as the Trojan was mail-based. The system's firewall blocked the HTTP reverse tunnel, but not the key logger, which used SMTP for transport. The HMI was on the enterprise network for the regional government. Multiple Internet connections in different agencies allowed both Web and email access from the HMI.

In 2005, an onshore oil production field SCADA system in the southwest U.S. was infected with the blaster worm, which infected all of the workstations and servers and which apparently entered the SCADA system via a network connection with the corporate WAN. Fortunately, the system remained functional, and AV tools were used by corporate IT personnel to remove the infection.

In 2011, the SCADA system of the city water utility in Springfield, Illinois, was hacked. Hackers gained remote access to the system, causing the system to turn a prime mover (pump) on and off repeatedly, leading to the burnout of the pump. On November 8, 2011, a water district employee detected the problems in the SCADA system, but forensic evidence suggests that hackers may have accessed the system as early as September 2011. As best it could be determined, the cyberattack was launched from an IP address in Russia and gained access by first hacking into the network of a software vendor that makes the SCADA system used by the utility. Usernames and passwords were stolen from a control system vendor by the hackers, who used them to access the water utilities network.

In 2003, two domestic electric power transmission companies had their SCADA/EMS systems disabled as a result of the SQL slammer worm being released onto the Internet. As we have mentioned previously in the book, the systems that comprise the Internet are the same telecommunication systems of the phone

companies and the "private" networks that they use to supply leased telecommunication services. One utility had leased frame-relay service to communicate with their RTUs. This service was run over an ATM network that was also carrying Internet and telephone traffic. The worm created such a bandwidth overload that the frame-relay service failed, causing loss of communication with the RTUs. The second utility was impacted by the same event, although in a different manner. A server on the utility's control center LAN running SQL was not patched. The worm apparently migrated through the corporate networks until it finally reached the critical SCADA network via a remote computer through a VPN connection. The worm propagated across the SCADA LAN to all computers and servers and effectively shut down the SCADA system.

These represent a small portion of the documented IACS cybersecurity incidents and attacks that have happened since 2000. There were some events in the 1990s, as well, but not as many in that decade, when IACS systems tended to be isolated and less vertically integrated into corporate business applications and systems. The point is that even if your SCADA system has never been attacked (at least that you know of), many such systems are being probed and attacked and yours could be next on the list. Certain major nation states, such as Russia and China, appear to be making a continuous and sustained effort to gain clandestine footholds in any IACS that controls an essential industrial process. Such a compromise may not have any impact on the current system operation, but if not identified and removed these backdoors would allow them to initiate malicious activities at some future point in time.

16

Commercial product vulnerabilities

All commercial operating systems and application software has been (or will be) the target of hackers looking for exploitable vulnerabilities. Essentially, most, if not all, commercial software, including operating systems and applications, has vulnerabilities. The problem is just as serious with open-source operating systems and applications. All of the major vendors, from Microsoft to Google, have produced software with vulnerabilities. We have already discussed the sorts of factors that introduce exploitable vulnerabilities, either remotely or locally exploitable. Vulnerabilities differ in their severity, the worst being vulnerabilities that enable the execution of malicious code. But other vulnerabilities may enable the bypassing of account security, enable a denial of service attack, or enable privilege escalation. As was previously mentioned, the MITRE organization has created a scoring system and database for identified vulnerabilities. The CVE™ (common vulnerability enumeration) list was begun in 1999 and is maintained and updated as new vulnerabilities are identified. A gentleman named Serkan Ozkan has taken the time and effort to put the MITRE data into a Web site with a large number of searching/sorting options: https://www.cvedetails.com.

On the CVE Details Web site, Mr. Ozkan and his associates have provided a summary page (fig. 16–1) that lists the various product manufacturers and sorts them by their overall number of reported software vulnerabilities. As that first screen shows, Microsoft is (and has been) the leading supplier of software with vulnerabilities, beginning with the earliest versions of Windows. But other major vendors with lots of exploitable product vulnerabilities include Oracle, IBM Google, Adobe, Apple, and Cisco. And the list also includes several Linux distributions and their known vulnerabilities. The point being that all software of any complexity will potentially have exploitable vulnerabilities, and new ones will be discovered periodically, and so you will need to be vigilant about patching and upgrading. But the whole point of assigning a point rating score to each CVE (0.0 to 10.0) is so that you can focus on the really dangerous vulnerabilities. We have already discussed the need for a patching policy and associated procedures and the possibility of using the CVE score values as a basis for elevating (or diminishing) the importance and urgency of patching newly identified vulnerabilities.

One of the very useful aspects of the CVE Details Web site is that it allows you to look at vulnerabilities on a product basis, not just an overall manufacturer basis.

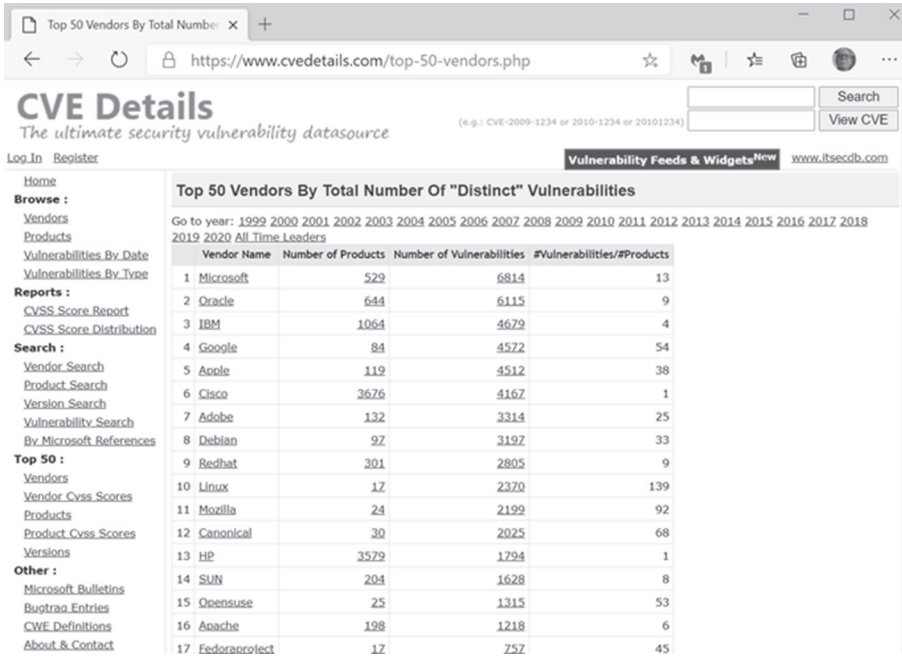


Fig. 16–1. The CVE database Web site based on MITRE data

If you are running Windows 2000 or ME or XP in some of your SCADA computers or servers, then you can search for vulnerabilities at that level. In figure 16–2, you can see a search for vulnerabilities specifically in Microsoft XP. For each entry in the resulting table, you are provided with a score value, but also with a lot of additional useful information including the vulnerability type (what the exploit does), the exploit access method (local or remote), and the difficulty level to the attacker to exploit the particular vulnerability. Those factors actually contribute to the score value for the vulnerability. It is also useful to be able to select only CVEs with a CVSS score greater than a specific value (e.g., 6.0 or greater), as those are the more serious vulnerabilities.

With a CVE ID number, you can perform a Google search and find the corresponding Microsoft (or whoever the vendor is) security bulletin that addresses that particular vulnerability. Microsoft maintains an online list of security bulletins that provide a general overview (Executive Summary) of the vulnerability, the software it impacts (e.g., XP, Windows 10, x64 version of Server 2008, etc.), the severity of the vulnerability, and also recommendations on patching the vulnerability and what updates are needed to fix the problem. Figure 16–3 shows an example security bulletin for MS12-052 (in their numbering scheme, the first number is the year and the second number is just consecutively assigned as new vulnerabilities are reported), which deals with a problem in Windows remote desktop. Security bulletins and advisories can be found online at: https://docs.microsoft.com/en-us/security-updates/

Fig. 16–2. CVE vulnerability listing for Windows XP

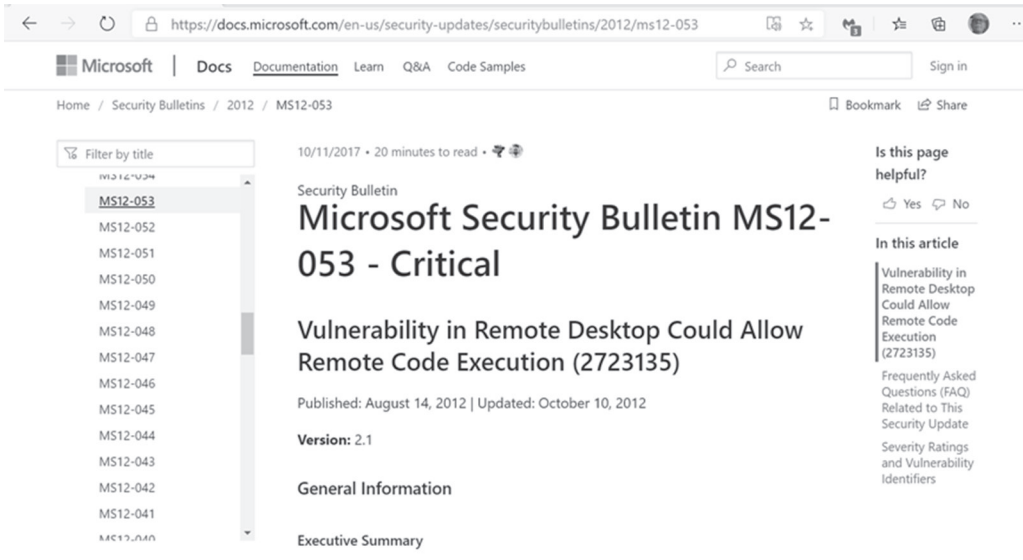


Fig. 16–3. Microsoft security bulletin Web site—example bulletin

The CVE Details Web site sorts all of the MITRE data and organizes it first by vendor and then by product. Just as Microsoft has identified vulnerabilities in its various Windows versions, Linux distributions are likewise tracked. Redhat

(purchased by IBM in 2019) has been a major player in the Linux world—initially, with its Fedora distribution and, more recently, with Enterprise Linux (which is used by IBM for products such as its QRadar SIEM product). You can search for CVEs associated with Enterprise Linux and with a score above a given level (e.g., 5.0). Figure 16–4 shows a search of the CVE Detail database for Enterprise Linux vulnerabilities. It is worth noting that on the Windows XP listing in Figure 16–2, there were only two pages of entries for a total of 71 vulnerabilities. Compare this to the fourteen pages of entries for a total of 662 vulnerabilities for Redhat Enterprise Linux. Some would argue that this is because the source code for Linux is readily available and so there are lots of researchers identifying issues.

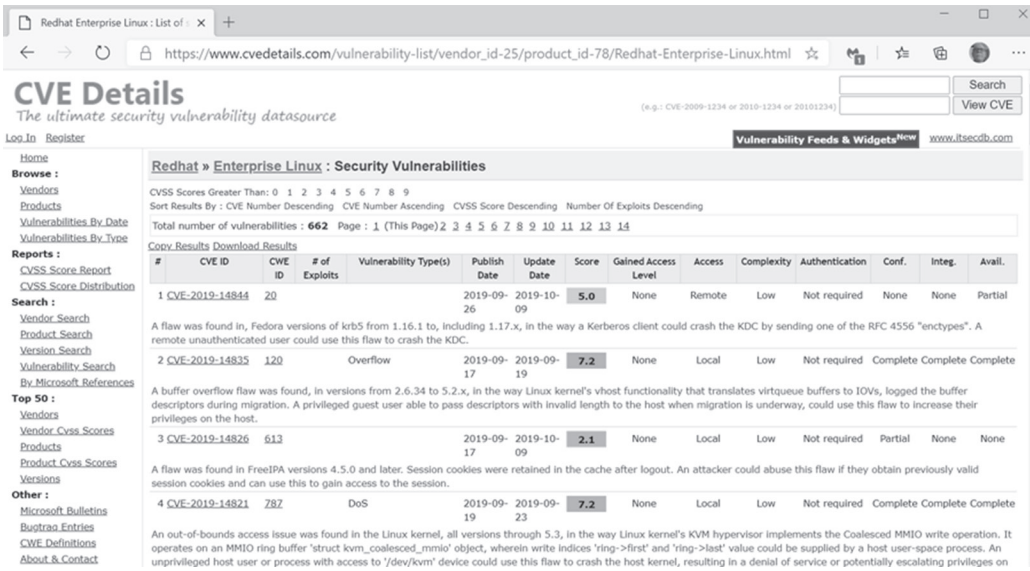


Fig. 16–4. The CVE Details Web page for Enterprise Linux vulnerabilities

NIST (National Institute of Standards and Technology) has a similar, related database called the National Vulnerability Database (NVD). It also lists identified vulnerabilities (and uses the same CVE numbering scheme as does MITRE) and applies the same form of CVSS scoring to each vulnerability. The NVD can be found online (https://nvd.nist.gov). One useful feature of this site is that it always maintains a list of the most recent 20 scored vulnerabilities, and so it is a good source to check to see if anything impacting your SCADA system has been identified recently (fig. 16–5). The following is an example of the summary information provided for each vulnerability:

> CVE-2019-4650 — IBM Maximo Asset Management 7.6.1.1 is vulnerable to SQL injection. A remote attacker could send specially-crafted SQL statements,

which could allow the attacker to view, add, modify or delete information in the back-end database. IBM X-Force ID: 170.
Published: June 26, 2020; 10:15:10 AM -04:00

If you require more details about a particular vulnerability, the summaries often include a hyperlink that takes you to a Web page with additional technical detail regarding the vulnerability. There may also be references to other online advisories, solutions, and tools associated with this particular vulnerability.

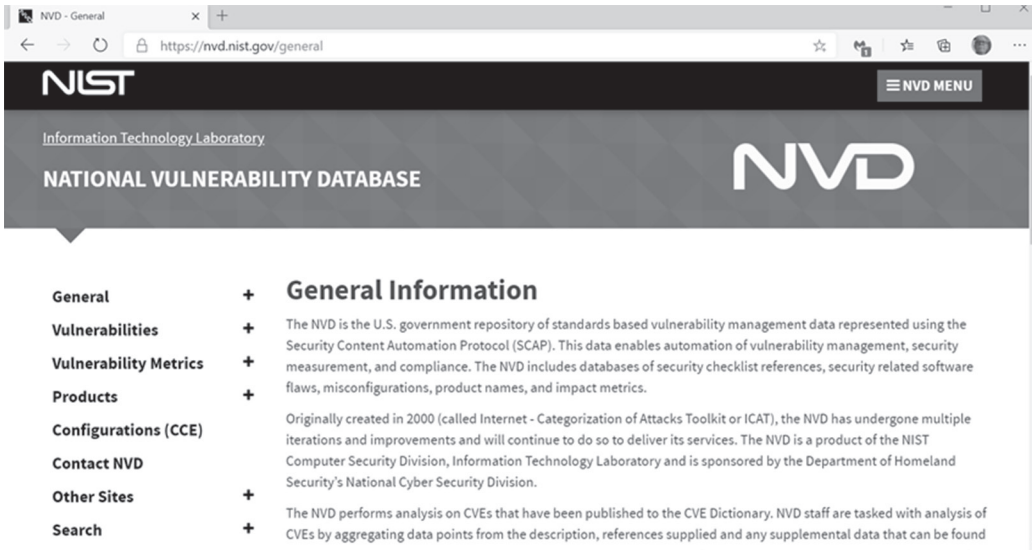


Fig. 16–5. NIST National Vulnerability Database (NVD)

Another great value in understanding the CVE naming and scoring scheme is that most manufacturers have linked their support portals and internal bug identification schemes to the CVE system, so that if a CVE corresponds to an IBM product or an Apple product or a Cisco product, the Google search of the CVE will usually turn up a link to that vendor's support page for that particular vulnerability (fig. 16–6). That means that, to a degree, the MITRE (or CVE Details) CVE Web site gives you a starting point for identifying any existing or newly discovered vulnerabilities in the products used to construct your SCADA system.

As part of maintaining your SCADA system's cybersecurity, it is important that you monitor credible sources of information about new threats and vulnerabilities, so that you can assess this information and make any necessary adjustments to your security posture. At a minimum, you should check the MITRE, NIST, and CVE Details Web sites to see if there are new vulnerabilities with a high score that are applicable to your SCADA system platforms. This is important because since

your SCADA system can't (or should not be able to) receive automatic patches and updates due to lack of Internet connectivity, you will need a periodic procedure for identifying if patches or updates are available (to fix new/newly discovered vulnerabilities) for any of the software products used in your SCADA system.

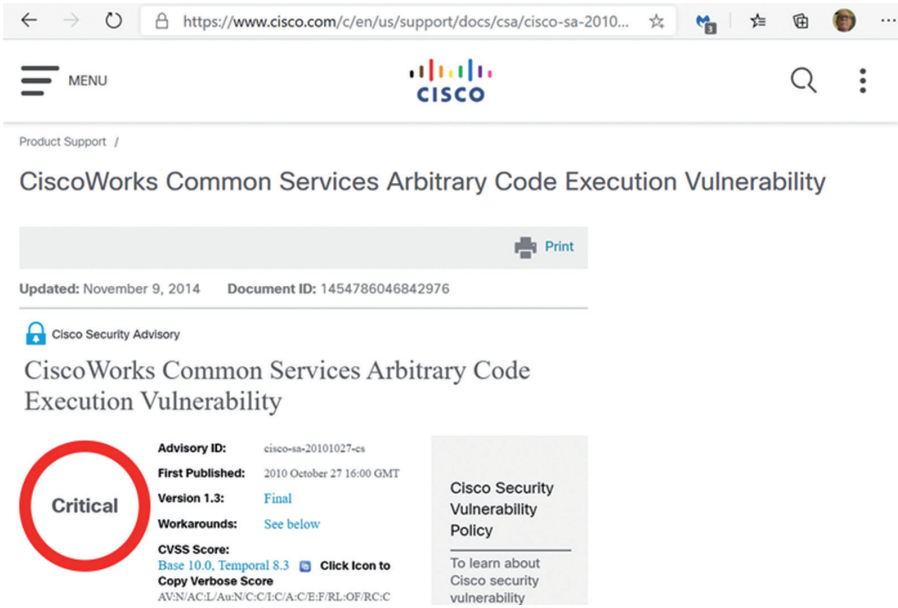


Fig. 16–6. Cisco's product support site for CVE assessment & support

Appendix A

U.S. Department of Energy's "21 Steps to Improved SCADA Security"

Shortly after the initial recognition by DHS of the vulnerability of the national infrastructure, DOE made an effort to provide some initial guidance and basic recommendations for securing automation systems (specifically, SCADA) by issuing a bulletin with twenty-one actions that could be taken to greatly improve the security of a typical SCADA system. These recommendations are still as valid today as they were when initially issued and thus are worth reviewing:

1. **Identifying all connections to SCADA networks**

   Comment— Connections are potential attack pathways, and so you want to be aware of any and all of them so that you can elect to remove them, defend them or at least monitor them.

2. **Disconnecting unnecessary connections to the SCADA network**

   Comment— See comments above.

3. **Evaluating and strengthening the security of any remaining connections to the SCADA network**

   Comment—This book has discussed protective and detective measures that can be applied to both IP-based network connections and to asynchronous serial communication links.

4. **Hardening SCADA networks by removing or disabling unnecessary services/ports**

   Comment—It is important to reduce the available attack surface in each computer, PC, and server; hardening and blocking unnecessary TCP and UDP ports is one aspect of hardening, so is eliminating unnecessary peripherals and applications and services.

5. **Avoiding reliance on proprietary protocols to protect the system**

   Comment—Vendor proprietary protocols are fading away and being replaced by IP-based protocols. But using proprietary protocols poses a double-edged sword: you assume an adversary can't learn about the protocol and attempt to abuse it for an attack (not a good assumption), but you can't monitor it with detective mechanisms (e.g., NIDS, firewalls) because they won't understand the protocol.

6. **Implementing security features provided by device and system (SCADA/computer) vendors**

   Comment—In many cases, operating systems for computers have a range of local policy settings that can be adjusted to enhance their cybersecurity. File systems can be encrypted and set for restricted access. User accounts can be access-limited. Likewise, many smart devices (e.g., switch, firewall, router) have settings that improve their cybersecurity such as selecting ssh or https versus telnet for remote administration.

7. **Establishing strong controls over any medium that is used as a backdoor into the SCADA network**

   Comment—Contrary to the proposed recommendation from the DOE, we would suggest eliminating any backdoors into your SCADA system. By definition, a "backdoor" is a pathway that is potentially not authorized or not conventional, and you may not be aware of everyone who is aware of the existence of the backdoor (e.g., former employees of your SCADA system vendor).

8. **Implementing internal and external intrusion-detection systems (IDS) and establishing 24-hour-a-day incident monitoring**

   Comment—A combination of HIDS, NIDS, and SIEM technologies provide a comprehensive means for detecting cyberattacks and suspicious or malicious user activity. But, like any "burglar alarm," there needs to be a means for alerting personnel when something is detected and for initiating an appropriate response, regardless of time or date.

9. **Performing technical audits of SCADA devices and networks and any other connected networks to identify security concerns**

   Comment—A good starting (or restarting) point for any cybersecurity program is to conduct a self-assessment, including an audit of digital assets and networks, so that you can perform a gap analysis and identify and rank the necessary actions to achieve adequate cybersecurity. Pathway analysis - determining all possible means through which a cyberattack could be instigated, is an essential part of this activity.

10. **Conducting physical security surveys and assessing all remote sites connected to The SCADA network to evaluate their security**

    Comment—As has been mentioned, remote connections should be protected by VPN technology or link encryption, if possible. Plus, if the physical and cybersecurity of the remote site(s) cannot be guaranteed, then the SCADA end of remote connections should include both a firewall and a NIDS sensor to monitor for malicious traffic or cyberattacks.

11. **Establishing SCADA "Red Teams" to identify and evaluate possible attack scenarios**

    Comment—An important aspect of an effective cybersecurity program is having an incident response plan and associated procedures, as well as training that includes testing various attack scenarios to validate that response procedures and processes are effective. Professional third-party cybersecurity assessment organizations exist to perform Red Team assessments, but possibly consider having your Corporate IT organization perform that role.

12. **Defining clear cybersecurity roles, responsibilities, and authorities for managers, system administrators, and users**

    Comment—Part of creating a cybersecurity program is the definition of the roles and responsibilities of various stakeholders and personnel. This is both as regards to implementing and maintaining the program and as regards to the response to cyber incidents.

13. **Documenting network architecture and identifying systems that serve critical functions or contain sensitive information that require additional levels of protection**

    Comment—The boundary of your SCADA system, and the cybersecurity protections put in place to secure that system, need to include any essential subsystems or support systems without which the SCADA system cannot function. The same sorts of technical measures should be taken to protect those systems from compromise: HIDS, hardening, whitelisting, etc.

14. **Establishing a rigorous, ongoing risk management process**

    Comment—No cybersecurity program will be perfect or will be effective forever without evolving. You must presume that what was good enough today might not be adequate next year. So, an annual review of the evolving threatscape should be part of your program, and part of that would be a reassessment of the threat actors and their capabilities.

15. **Establishing a network protection strategy based on the principle of defense-in depth**

    Comment—Defense in Depth is always a good strategy, as a single compromise of a defensive measure should not expose the entire SCADA system to compromise. The use of multiple layers of defense and detection creates an effective barrier against attacks and should provide an opportunity to react and respond prior to any debilitating actions on the part of the attacker.

16. **Identifying clear cybersecurity requirements**

    Comment—This recommendation touches many aspects of your cybersecurity program, but two are most essential: (1) to clearly

communicate to all personnel—including contractors, vendors, and partners—the security policies, procedures, and objectives that apply to them; and (2) to ensure that management is on board with the stated objectives of the cybersecurity program and that they communicate the importance of following cybersecurity policies to all employees and contractors.

17. **Establishing effective configuration management processes**
    Comment—As part of your cybersecurity program, you need to perform hardening (including making applicable local policy and account settings) on your various PCs, computers, and servers (and even VMs), and thereafter you need to ensure that you preserve these baseline configurations so that they are available, if needed, for restoration and recovery purposes. Then, as you install patches and updates and make changes to any of these devices, you need to make new baselines and backups. Also, part of configuration management is controlling who has access to both make/store and extract/apply these baseline configurations. Plus, you may need to document an audit trail of changes so that you know what was done, when, and why. This can be useful when the question arises of what systems have received patches for CVE-XX-YYY.
18. **Conducting routine self-assessments**
    Comment—Most cybersecurity experts agree that an annual review of your program and procedures is essential for ensuring that it remains effective against the evolving threatscape. As it is all too easy for people familiar with the program to miss important deficiencies, it is often recommended that an outside party (a fresh pair of eyes) be brought in to perform or at least support the assessment effort.
19. **Establishing system backups and disaster recovery plans**
    Comment—Your plan should assume that the worst-case scenario could happen and plan for what to do. And, of course, you should have an increasing set of response plans based on what could actually/realistically happen, starting at a minor probe to a full-blown compromise. It is not good enough to just write up a bunch of plans. You need to rehearse them (at least doing a tabletop review) periodically to ensure that they will work as expected and can be performed and so that personnel know their roles and responsibilities in each scenario. Occasional surprise declarations of cyber events can be used to keep people on their toes and to provide both practice and training.
20. **Establishing expectations for cybersecurity performance and holding individuals accountable for their performance**
    Comment—Experts recommend that all personnel be made aware of the importance of maintaining adequate cybersecurity and

why required procedures and processes are essential. If there are consequences for failing to follow required processes and procedures (and there probably should be), these potential consequences should be clearly communicated as well.

21. **Establishing policies and conducting training to minimize the likelihood that organizational personnel will inadvertently disclose sensitive information regarding SCADA system design, operations, or security controls**

    Comment—Every organization should have an information classification and protection policy because today's corporations have information of a technical nature as well as information of a business nature—plus, of course, information of a personal nature—and each requires some level of protection. We will ignore regulations such as HIPPA and SOX because this book deals with SCADA cybersecurity. Clearly, there is information about the SCADA system that needs to be kept confidential because its disclosure could aid an adversary in launching a cyber (or even physical) attack. Clearly, information such as user IDs and passwords needs to be protected, but so do network drawings and things like the firewall rules and local user policy settings. Your SCADA organization ought to specifically identify categories of information, both stored in physical form (e.g., drawings) and electronic form (e.g., firewall rule set files), and determine what sorts of protections are necessary. This could include locked cabinets for information in physical form and the use of encryption for information stored in electronic form. Part of the information protection policy should be identification of personnel (by job role or title) authorized to access and delete/destroy sensitive information and authorized to edit/change sensitive information.

Appendix B

NERC CIP—Recommendations for Electric Utilities

This appendix contains key technical and operational points related to SCADA system security, as recommended by the North American Electric Reliability Council (NERC). The points extracted from the current requirements break down the various aspects of physical and electronic security and the actions that need to be taken to secure a SCADA system, develop a security program, and maintain an acceptable level of ongoing security. Since its issuance, the original *1300* document was replaced by a set of individual "Critical Infrastructure Protection" standards (originally, CIP-002-1 through CIP-009-1) that essentially take the individual sections of 1300 and make them separate items. Although NERC specifically intended these recommendations for the electric utility industry, these recommendations, in whole or in part, are being widely adopted by standards organizations in other industrial segments and therefore are well worth review, regardless of the industry. In its initial issuance, these standards were focused on electrical power transmission and generation management SCADA/EMS systems, but over time, the CIP standards were extended to include DCS systems and other IACS technology in generating plants, substations, and other Bulk Electric System critical assets. Since their initial issuance, the CIPs have been expanded to cover things not part of the original set of recommendations, such as addressing the supply chain threat. As of 2020, there are 13 individual CIP standards, with multiple revisions and updates for most. There are also additional changes being considered and reviewed. It is worth noting that the original standard 1300 was issued in 2004, and the standards are still evolving as the understanding of threat actors has likewise evolved. One reason for pointing out these standards is that they are not static and are constantly being revised to meet the evolving capabilities and techniques of the various domestic and international threat actors. The CIP standards can be found on the NERC website (https://www.nerc.com/Pages/default.aspx).

The following is a summary of each of the current CIP standards, excluding CIP-001, which is administrative in nature:

CIP-002-5.1a Cyber Security—BES Cyber System Categorization

This standard requires you "to identify and categorize Bulk Electric System (BES) cyber systems and their associated BES cyber assets for the application of

cyber security requirements commensurate with the adverse impact that loss, compromise, or misuse of those BES cyber systems could have on the reliable operation of the BES. Identification and categorization of BES cyber systems support appropriate protection against compromises that could lead to mis-operation or instability in the BES."

Associated actions—You must identify each critical asset, categorize the asset by function and basis (is it digital/cyber), prioritize how serious the consequences to the BES would be with its compromise or loss, and characterize the overall relationship or operating dependency between the asset and your facility. Essentially, you must create a cyber (digital) asset database with sufficient characterization information for each asset.

CIP-003-6 Cyber Security—Security Management Controls

This standard requires you "to specify consistent and sustainable security management controls that establish responsibility and accountability to protect BES cyber systems against compromise that could lead to mis-operation or instability in the BES."

Associated actions—You must implement consistent and sustainable security management controls (e.g., countermeasures) to protect all identified critical cyber assets from compromise, misoperation, or instability. Issues such as cybersecurity policy, leadership buy-in, site-specific exceptions, information protection, access control (logical), change control, and configuration management are all included in this CIP.

CIP-004-6 Cyber Security—Personnel & Training

This standard requires "the minimizing of risk against compromise that could lead to mis-operation or instability in the BES from individuals accessing BES cyber systems by requiring an appropriate level of personnel risk assessment, training, and security awareness in support of protecting BES Cyber Systems."

Associated actions—You must ensure that all personnel with authorized access (physical and logical) to critical cyber assets have an adequate degree of personnel screenings, risk assessments, employee training, and attendance at cybersecurity awareness programs. You also need to maintain a list of credentialed access lists, including service providers and contractors. You must also document, track, review, and update such personnel training and training programs annually.

CIP-005-5 Cyber Security—Electronic Security Perimeter(s)

This standard requires you "to manage electronic access to BES cyber systems by specifying a controlled Electronic Security Perimeter in support of protecting BES cyber systems against compromise that could lead to mis-operation or instability in the BES."

Associated actions—This set of requirements focuses on creating, monitoring, and maintaining your "electronic" perimeter and on actions to address vulnerabilities created by allowing remote access. The "electronic" perimeter that

encompasses all critical cyber assets should be protected, and all access pathways through the perimeter must be identified and secured. Applicable technical countermeasures include remote session encryption (VPN), multi-factor authentication, anti-malware updates (AV and endpoint protection), patch updates, and using role-based access for restriction user accounts.

CIP-006-6 Cyber Security—Physical Security of BES Cyber Systems

This standard requires you "to manage physical access to BES cyber systems by specifying a physical security plan in support of protecting BES cyber systems against compromise that could lead to mis-operation or instability in the BES."

Associated actions—These requirements deal with creating, monitoring, and maintaining a physical security perimeter and with implementing a physical security program. The goal is to identify the physical security zone(s) and implement preventative controls (physical, technical, and administrative) aimed at protecting and controlling access to cyber assets within the security zones. The standard requires the creation of a physical security plan, protection of physical access control systems, protection of electronic access control systems, physical access controls, physical access monitoring, physical access logging and log retention, plus periodic maintenance and testing.

CIP-007-6 Cyber Security—System Security Management

This standard requires you "to manage system security by specifying select technical, operational, and procedural requirements in support of protecting BES cyber systems against compromise that could lead to mis-operation or instability in the BES."

Associated actions—This requires that you create, implement, and maintain processes and procedures for securing systems for both critical and non-critical cyber assets. This also means documenting security measures, including records of test procedures, ports, and services, security patch management, and malicious software prevention.

CIP-008-5 Cyber Security—Incident Reporting and Response Planning

This standard requires "mitigation of the risk to the reliable operation of the BES as the result of a cybersecurity incident by specifying incident response requirements."

Associated actions—Security incidents related to any critical cyber assets must be identified, classified, responded to, and reported in a manner deemed appropriate by NERC. You will want to create an incident response plan that should include the actions, roles, and responsibilities of those involved, as well as details of how incidents should be handled and reported to governing bodies. This plan will need to be updated annually and tested for applicability.

CIP-009-6 Cyber Security—Recovery Plans for BES Cyber Systems

This standard requires that you "recover reliability functions performed by BES cyber systems by specifying recovery plan requirements in support of the continued stability, operability, and reliability of the BES."

Associated actions—You must develop and test and train personnel on recovery plans that adhere to disaster recovery best practices. This standard require you to establish recovery plans, change control processes, backup and restoration processes, and testing of backup media to verify it is viable.

CIP-010-2 Cyber Security—Configuration Change Management and Vulnerability Assessments

This standard requires "preventing and detecting unauthorized changes to BES cyber systems by specifying configuration change management and vulnerability assessment requirements in support of protecting BES Cyber Systems from compromise that could lead to mis-operation or instability in the BES."

Associated actions—You must establish a change management process and associated procedures, including testing and verification of changes, plus processes for escorted access and vetting of vendor maintenance personnel.

CIP-011-2 Cyber Security—Information Protection

This standard requires you "to prevent unauthorized access to BES cyber system Information by specifying information protection requirements in support of protecting BES cyber systems against compromise that could lead to mis-operation or instability in the BES."

Associated actions—You must establish an information categorization policy and information protection procedures, including both physical and electronic information.

CIP-014-2 Cyber Security—Physical Security

This standard requires you "to identify and protect transmission stations and transmission substations, and their associated primary control centers, that if rendered inoperable or damaged as a result of a physical attack, could result in instability, uncontrolled separation, or cascading within an interconnection."

Associated actions—This requirement deals with putting in place physical security (fences, walls, gates, man-traps, bollards, etc.) measures that block unauthorized access to facilities by people and vehicles and which may also include checking vehicles for explosives and checking personnel bags and backpacks for weapons, explosives, or dangerous materials.

DEFINITIONS—Terms used in the CIP Standards

Cyber Assets: Those systems (including hardware, software, and data) and communication networks (including hardware, software, and data) associated with bulk electric system assets.

Critical Cyber Assets: Those cyber assets that perform critical bulk electric system functions, such as telemetry, monitoring and control, automatic generator control, load shedding, black start, real-time power system modeling, special protection systems, power plant control, substation automation control, and real-time inter-utility data exchange, are included at a minimum. The loss or compromise of these cyber assets would adversely impact the reliable operation of bulk electric system assets.

Bulk Electric System Asset: Any facility or combination of facilities that, if unavailable, would have a significant impact on the ability to serve large quantities of customers for an extended period of time, or would have a detrimental impact on the reliability or operability of the electric grid, or would cause significant risk to public health and safety.

Electronic Security Perimeter: The logical border surrounding the network or group of subnetworks (the "secure network") to which the critical cyber assets are connected, and for which access is controlled.

Physical Security Perimeter: The physical border surrounding computer rooms, telecommunications rooms, operations centers, and other locations in which critical cyber assets are housed and for which access is controlled.

Responsible Entity: The organization performing the reliability function, as identified in the Reliability Function table of the Standard Authorization Request for this standard.

Incident: Any physical or cyber event that:

- Disrupts, or could have led to a disruption of, the functional operation of a critical cyber asset, or
- Compromises, or was an attempt to compromise, the electronic or physical security perimeters.

Security Incident: Any malicious or suspicious activities that are known to cause, or could have resulted in, an incident.

Appendix C

Security Recommendations of the Instruments, Systems, and Automation Society and the American Gas Association

The ISA (International Society for Automation [formerly the Instrument Society of America]) is actively involved in attempting to develop guidelines for improving the security of automation systems, including SCADA systems, DCS, and PLC-based systems. As part of this effort, they have created and issued a technical report entitled "Integrating Electronic Security into the Manufacturing and Control Systems Environment" (ANSI/ISA-TR99.00.02-2004). This is a working document, subject to periodic review and modification, if required, which has been issued not as a standard, but rather as a set of recommendations (a Technical Report). The ISA's recommendations were developed by soliciting the input of industry experts and the end-user community (companies that employ computer-based automation systems). This document names vendors, end-users, and industry experts that participated in the development of the document.

A very important aspect of the creation of this report is the participation of people who are also active with other organizations attempting to develop equivalent standards and recommendations. To date, there are at least a dozen groups working, in parallel and independently, on standards and recommendations to achieve the same end result: cybersecure automation systems. These groups are listed in Appendix D.

The work being done by most of these groups is somewhat redundant—with a high degree of overlap and, in some cases, contradiction. In other instances, the work being done is targeted to a specific area of need, such as the AGA's recommendation for a light-weight encryption standard for secure serial communications to field devices (AGA Report No. 12 includes a specific section entitled "Low Latency Cryptographic Protection for SCADA Communications" [a separate discussion of this document follows]). In many cases, the general recommendations of these groups are essentially an amplification and detailed expansion of the initial recommendations by the NERC in 2003.

In the summer of 2005, an effort was made to bring all of these different groups together, to coordinate their work, establish lines of communication, and

standardize terminology. This first International Standards Coordination Meeting was co-sponsored by the Department of Homeland Security and is planned as an annual event. The ISA is already coordinating with many of the groups listed previously and has made an effort, in its technical report, to incorporate much of the work being done by these different groups.

The technical report clearly distinguishes cybersecurity issues, practices, and solutions appropriate in a classical IT environment from those that would be suited to a control system environment. This distinction is critical, since certain standard IT practices (e.g., taking a system out of service during off hours to install new patches and software updates) don't translate well into a process/production plant environment where control systems operate 24/7. The technical report also discusses the process of developing a security program and the process of getting management to buy in—for the effort and costs—by developing a business case. This is where the involvement of actual end users is invaluable, as they are aware of the corporate blockades that must be surmounted and are aware that, because you can't circumvent them, they have to be addressed. The report recognizes the business realities of obtaining funding for (cyber)security programs and acknowledges that the goal should be adequate electronic security, rather than the creation of the perfect or the most sophisticated security program. Part of this is the process of doing a security assessment, which includes weighing the possibility and consequences of various threat agents and determining the cost-justifiable range of countermeasures that ought to be implemented.

The major topics addressed in the technical report include:

- Development of a security program with a business case justification
- Enumeration of risks and their probabilities and costs
- Performing a risk assessment and gap analysis
- Selecting appropriate countermeasures for the key vulnerabilities
- Implementing the selected counter measures
- Developing and executing test plans
- Establishing and maintaining an operational security program
- Setting up a change-management program
- Developing disaster recovery and response plans

The report addresses a wide range of topics, but it does not offer a how-to solution for most of them. Rather, the report generally lists the issues and the tasks that should be performed, but leaves the details of implementation to the reader. In some instances, where a suitable ISO standard exists, the report will direct the reader to that applicable ISO standard. The report does include a range of examples and templates for documentation, such as summarizing a plant's critical control/computer/automation assets, defining and assessing risk, and writing a security policy.

Because the report covers all types of computer-based automation, including continuous, Batch, and discrete manufacturing automation, it addresses critical

assets that would not typically apply in most SCADA applications. A bar-code scanner might prevent a truck from being loaded with boxes at a discrete manufacturing plant, but it would be a rare SCADA system that depended on such a device. By contrast, the report excellently describes the differences between an insider and an outsider as a possible threat agent. It also offers a very good general outline of the special considerations that make automation cybersecurity different from conventional IT cybersecurity:

- The risks involved include loss of life, facilities, and plant equipment; endangering the public/public health; and damage to product.
- It is not normally corporate information that must be protected (although that is a factor in some SCADA systems), but the plant and process. The fact is that specialized automation devices (e.g., PLCs and RTUs) may be more critical than the central SCADA server/computer.
- Automation systems require nearly perfect availability and may control processes that are unstable without constant supervision (this is less the case in SCADA applications than in process control applications, but still true to some degree).
- Control systems (and the processes they control) often involve time-critical responses that preclude making procedural changes that introduce delays (e.g., having an operator enter a password before a command is sent). However, again, this is less of a factor in SCADA applications.
- Communications may involve proprietary protocols and networks, or at least non-IP-based communications, such that common IT tools (e.g., an Ethernet packet sniffer) are ineffective or useless.
- Information integrity may be more critical in a control system where a bad value could cause a process upset or plant shutdown.
- Software maintenance and change management may—and really should—involve the control system vendor, since patches to the underlying operating system software could affect the operation of the control system software. Such updates and patches should be applied and tested by the SCADA system vendor and should not be applied until approved by the vendor.

Recommendations of the AGA

AGA Report No. 12, "Cryptographic Protection of SCADA Communications—General Recommendations," addresses the issue of securing the existing communication channels in a SCADA system between the central host and the field-based remotes. The basic assumption of the AGA is that these low-bandwidth communication channels, with their slow-speed, serial, asynchronous communications

protocols, are probably not going away any time soon and thus need to be protected. The subject of hijacking an RTU has been discussed in great detail in the present book; suffice it to say, this is a real concern. As has been mentioned elsewhere in this book, there are link encryption devices suitable for placement in an RS-232/422/485 serial communication circuit and encrypting the messages flowing in both directions. These are external devices, and they cannot be directly placed on an analog telephone circuit but between the RTU/PLC and the modem used to connect to such a circuit. This makes them unusable (or very difficult to use) for older RTUs with an integral modem. The AGA proposed an encryption algorithm that could be implemented with minimal memory and CPU requirements (what is meant by describing the algorithm as "lightweight"), which could potentially be programmed into RTUs and PLCs and other smart devices (and of course the SCADA masters as well), thus eliminating the need for hardware modifications.

The AGA has recommended an encryption standard that could be implemented in commercial products and that is designed to be compatible with low-bandwidth/low-baud-rate protocols. The scheme proposed addresses mutual authentication, certificate authentication, session keys, and link encryption. The actual standard has four sections (12-1 through 12-4), and each takes a slightly different view of SCADA communications security, including security over digital networks, although, to date, only the initial two sections have been completed.

Of course, this cannot be retrofitted directly into the programming and hardware of older RTUs. It would require external hardware placed in front of the RTU on the communication circuit. However, the latest RTUs, with Pentium-class or newer CPUs, could potentially be reprogrammed to incorporate all of these functions and features. Unfortunately, since its release, not many SCADA or RTU/PLC vendors have elected to adopt and implement the recommended standard, either as add-on hardware or directly within their RTU products.

Furthermore, the central host end of the communications circuit has to support these same functions. Again, this could be handled either as an external hardware fix or as a software enhancement to the SCADA system itself. The major accomplishment of the AGA report (and the proposed encryption standard) is that it is supposedly designed to function within the real-world constraints of typical SCADA polling channels. In fact, the encryption scheme was renamed SLS—SCADA Link Security, and several researchers evaluated the proposed standard. The standard has four parts, and Part 2 describes the actual serial SCADA protection protocol (SSPP). The Pacific Northwest National Laboratory published a performance test on Part 2 of the standard in 2007, based on the DNP3 protocol. Unfortunately for the standard, IP to the field and IP-ready RTU and PLC product began gaining traction around the same time, and, as a result, few if any products resulted from the standard. Once an IP-based WAN is used for field communications, there are numerous encryption possibilities available to protect SCADA communications.

Appendix D

Industry and Government Security Recommendations

The following is a list of various governmental, industry, and international organizations that are developing or have published security recommendations related to SCADA systems.

- Instruments, Systems, and Automation Society (ISA)—TR1, TR2, and SP99 Standards (now ANSI/ISA 62443) plus their ISAsecure© program for smart devices
- Process Control Security Requirements Forum (PCSRF)
- International Electrotechnical Commission (IEC)—TC 57 and 65 plus 62443
- U.S. Department of Homeland Security (DHS)—Process Control Systems Forum (PCSF)
- Institute for Information Infrastructure Protection (I3P)—SCADA Initiative
- U.S. Department of Energy (DOE)—National SCADA Test Bed
- U.S. National Institute of Standards and Technology (NIST)—Industrial Control Systems Test Bed
- Control Systems Security Center—INL
- Center for SCADA Security—SNL
- U.S. CERT Control Systems Center
- International Council on Large Electric Systems (CIGRE)
- Institute of Electrical and Electronic Engineers (IEEE)—Power Engineering Society (PES)
- North American Electric Reliability Council (NERC)—Standard 1200, Standard 1300 plus CIP-001 through CIP-014
- Electric Power Research Institute (EPRI)
- Process Control Systems Cyber Security (PCSCS) Forum
- Chemical Industry Data Exchange (CIDX)
- American Petroleum Institute (API) 1164
- American Gas Association (AGA) Report No. 12
- American Water Environment Federation (WEF)

- International Standards Organization (ISO)
  - ISO 17799
  - ISO 15408
  - ISO/IEC 27001 and 27002
- Applicable NIST 800-series special publication documents:
  - SP 800-48: Wireless Network Security
  - SP 800-41: Firewalls
  - SP 800-40: Security Patches
  - SP 800-31: Intrusion Detection
  - SP 800-53: Cyber Security Controls
- American Water Works Association (AWWA)
- Nuclear Regulatory Agency (NRC) RegGuide 5.71

As can be seen, there is and has been a lot of ongoing activity and a lot of work being done by various groups that may end up replicating the efforts of others. Out of all of this activity, it can be presumed, will emerge new standards and recommendations.

It is worth specifically noting the excellent work being conducted by NIST and that NIST and their complete set of Special Publication 800 series documents are a good source for a wide range of recommendations and technical information on a long list of cybersecurity topics.

Appendix E

SCADA System Security Assessment Checklists

The following checklists are intended to assist an organization with a SCADA system, or those responsible for assuring SCADA system security, with the process of collecting and evaluating the information relevant to cybersecurity. Many of the checklists are based on the overall recommendations documented by NERC in their CIP standards.

Even though multiple, separate tables are shown for the variety of security-related information to be tracked and recorded, note that this information (as with most of these tables) will in actual practice probably be implemented in an electronic form (spreadsheets or database applications) in which the multiple tables can be integrated into a single compound form or implemented as linked database tables.

These checklists are provided only as examples of the type of information to be recorded and tracked. The actual details and mechanisms used will vary by organization.

CATEGORY: Security policies and procedures

WRITTEN POLICIES

No. assigned	Description/requirement	Initially issued on	Manager responsible and title	Current revision no.	Last updated on	Next review date
POL-001	Corporate policy on information management, protection, disclosure, and confidentiality,					
POL-002	Use of company systems and networks					
POL-003	Passwords, user accounts, and account maintenance					
POL-004	Telephone usage including personal cell phone					
POL-005	Internet usage and prohibited Web sites					
POL-006	Software licensing policy					

POL-007	Portable computing policy					
POL-008	Personal computer equipment and software					
POL-009	Removable media policy					
POL-010	Email and IM usage and content					
POL-011	Facility access policy and electronic credentials					
POL-012	Background checks prior to and during employment					
POL-013	Monitoring of computer usage by employees					
POL-014	Handling and care of confidential information					
POL-015	Classifications for information protection and access					
POL-016	Downloading of data, software, or files					
POL-017	Virus scanning of computer equipment and media					
POL-018	Performance reviews and employment termination					
POL-019	Use or possession of personal electronic devices on company property					
POL-020	Wi-Fi- and Bluetooth-equipped devices					
POL-021	Spam, restricted email list, and email attachments					
POL-022	Remote access and VPN usage					
POL-023	Duplication of and removal of sensitive documents					
POL-024	Video surveillance and physical searches of persons, possessions, and office area					

The policies listed above are provided as examples and are not to be considered as a comprehensive list

INSTRUCTIONS: Create and maintain an accurate list of all published company policies related to information management, confidentiality, computer usage, account management, confidentiality, communications, Internet access, equipment and software validation and approval, baselines, requirements, standards, and anything associated with critical computing systems, networks, and associated regulations such as HIPPA and SOX.

A senior manager should be assigned responsibility for each policy, and policies should be reviewed and updated as/if needed, on an annual (or more frequent) basis. All written policies should include an audit log of changes and written approvals for issuance. Policies that relate to others should list those dependencies and relationships in a specific document section. Policies that have been identified as necessary, but not yet written or completed, should at least be enumerated in this list above and assigned both a unique ID number and a manager with the responsibility for policy completion. This list itself should be treated as an essential document and reviewed monthly, to schedule and organize impending policy reviews, or whenever a new or revised existing policy is generated.

DOCUMENTED PROCEDURES

No. assigned	Description/requirement	Initially issued on	Manager responsible and title	Dependencies	Last reviewed on	Next review date
PRC-001	Initializing and starting up the SCADA system after a total shutdown					
PRC-002	Transferring the SCADA system to the redundant backup equipment set					
PRC-003	Restoring SCADA system operation to the primary equipment set after a transfer					
PRC-004	Transferring operations to the alternative operations site and systems					
PRC-005	Restoring the primary operations site and systems after a transfer					
PRC-006	Making system configuration changes					
PRC-007	Loading and testing of system software patches and updates					
PRC-008	Loading and testing of application software changes and new software					
PRC-009	Modifying operational displays, reports, logs, and alarm settings					
PRC-010	Making a SCADA system backup copy					
PRC-011	Restoring the SCADA system from a backup copy					

PRC-012	Adding or deleting new user account or modifying existing account settings					
PRC-013	Installing and configuring a new RTU on the SCADA system					
PRC-014	Setting up, configuring, and testing an RTU prior to field installation					
PRC-015	Disposing of computer and electronic equipment, removable media, and printed documentation					
PRC-016	Issuing and revoking of digital certificates					
PRC-017	Issuing and revoking of electronic credentials and system accounts					
PRC-018	Periodic testing of physical access controls and monitoring equipment					

The procedures listed above are provided as examples and are not to be considered as a comprehensive list

INSTRUCTIONS: Create and maintain an accurate list of all published company procedures related to the procurement, installation, testing, certification, maintenance, and restoration of critical computer systems, communications subsystems, and critical application and system software. A senior manager should be assigned responsibility for each procedure, and procedures should be reviewed, tested, practiced, and updated as/if needed, on an annual (or more frequent) basis. All materials, information, and equipment assets necessary for the execution of a given procedure should be clearly and completely enumerated in the front section of the procedure. All written procedures should include an audit log of changes and written approvals for issuance. Procedures that relate to others should list those dependencies and relationships in a specific document section. Procedures that incorporate vendor-supplied documentation should clearly reference those related documents and, if possible, include actual excerpts. Procedures that have been identified as necessary, but not yet written or completed, should at least be enumerated in this list above and assigned both a unique ID number and a manager with the responsibility for policy completion. This list itself should be treated as an essential document and reviewed and updated quarterly or at any time a new or revised procedure is generated.

CATEGORY: Asset identification/inventory

EQUIPMENT ASSETS

No. assigned	Description/ requirement	Qty.	Manufacturer serial no./model	Inventory no.	Building ID	Room or area
EQ-001						

INSTRUCTIONS: Create and maintain an accurate list of all vital equipment and/or systems, to the server or major component level. This list itself should be treated as an essential document and reviewed and updated quarterly or whenever major equipment is added or removed. The organization must make determinations in regard to what types of equipment are considered as sufficiently important to be placed on this list and to what level of granularity the equipment should be identified. Equipment whose loss or failure will affect the ability of the organization to meet its operational requirements, or which will potentially endanger personnel or facilities by its loss, should be included on this list. A major reason for having and maintaining this list is to ensure that equipment on the list is appropriately considered when making decisions about necessary operational spare parts, service contracts, maintenance planning, and system redundancy. The information maintained on this list can be expanded to whatever level is most useful to the organization. Contacts and telephone numbers for service and support, storage location of replacement components, and other such information can be added to this list if such data are not already maintained in other databases. Linkages to other such databases can be added to this list as well.

COMMUNICATION ASSETS

No. assigned	Description/usage	Qty.	Vendor and reference ID	Type function	Building ID	Room or area
COM-001						

INSTRUCTIONS: Create and maintain an accurate list of all vital communication equipment such as routers, FRADs, LAN switches, firewalls, radio and microwave equipment, and modems. This is the electronic equipment

necessary for the communication links between the SCADA system and the field devices, other systems, and external and internal networks. This list should be treated as an essential document and reviewed and updated quarterly or whenever major equipment is added or removed. The organization must make a determination in regard to what level of granularity the equipment should be identified. Equipment whose loss or failure will affect the ability of the organization to meet its operational requirements, or which will potentially endanger personnel or facilities by its loss, should be included on this list. A major reason for having and maintaining this list is to ensure that equipment on the list is appropriately considered when making decisions about necessary operational spare parts, service contracts, maintenance planning, and system redundancy. The information maintained in this list can be expanded to whatever level is most useful to the organization. Contacts and telephone numbers for service and support, storage location of replacement components, and other such information can be added to this list if such data are not already maintained in other databases. Linkages to other such databases can be added to this list as well. It may be useful to include cross-references to the respective communication circuit (listed subsequently as "Electronic Access Points").

INFORMATION ASSETS

No. assigned	Description/title	Qty.	Responsible manager and title	Media	Building ID	Room or area
INF-001						

INSTRUCTIONS: Create and maintain an accurate list of all vital information and documentation, regardless of its medium. This is the information necessary for the support, operation, and maintenance of the SCADA system and related critical subsystems. This list itself should be treated as an essential document and reviewed and updated annually or whenever changes are made that affect this list. Information that would normally be on this list would include user's guides, operating and maintenance instructions, configuration settings, system databases, backup copies of all system software, and so forth. Any information, data, file, or document that would be needed to restore the system (or a critical subsystem) after a failure should be included on this list. Any such items necessary for the execution of a procedure (see the previous list entitled "Documented Procedures") should be included on this list. If materials are in secure storage or require additional levels of authority for access, this information should be added to the list. The purpose of this list is to identify and locate all critical information assets, so that they can be located when needed and so that suitable precautions can be taken for their storage and preservation. Any critical information that is normally maintained in a file or database table on a computer ought to have a corresponding documented procedure for its periodic backup and storage.

CATEGORY: Personnel management

AREA ACCESS RIGHTS

Employee ID	Employee name	Department	Supervisor	Telecom room	Server room	Control room	SCADA room

INSTRUCTIONS: Create and maintain an accurate list of employees who have been granted access rights to sensitive areas, and review this list regularly (on a biannual basis) and whenever an employee is added or job responsibilities are changed. Use different indicators to differentiate personal access rights and the right to bring others, who do not have access rights, into each area. The list of secured areas will be specific to an organization and may encompass multiple facilities. This list should be customized to match the specific requirements of each organization.

ELECTRONIC ACCESS RIGHTS

Employee ID	Employee name and account IDs per system	Department	Supervisor	Office PC	Remote telecom	IT servers	Email server	SCADA system

INSTRUCTIONS: Create and maintain an accurate list of employees who have been granted access rights to both business and operational computer systems, and review this list regularly (on a biannual basis) and whenever an employee is added or terminated or job responsibilities are changed. All user IDs granted to an employee (or contractor) should be listed, including those for company-provided PCs. The account designation should indicate the authorization level for the account, if multiple levels are defined. This list should be customized to match the specific requirements of each organization. If electronic credentials, such as ID cards and password tokens, are issued to employees, these should be noted on this list, and sufficient information should be provided to support the revocation procedure for those items. If temporary access rights are granted to contractors, the list ought to include a place to record the date when those rights are to be revoked.

SECURITY TRAINING & BACKGROUND CHECKS

				Basic security	Cybersecurity	
Employee ID	Employee name	Department	Supervisor	Date of last training	Date of last training	Date of last background check

INSTRUCTIONS: Create and maintain an accurate list of employees who have received training in cybersecurity, basic security, and social engineering techniques, and review this list regularly (on an annual basis) and whenever an employee is added or job responsibilities are changed. Additional security training may be required for selected job positions and when new threats are identified that require training augmentation. Also, track the performance and review of employment background checks (and rechecks) to ensure that personnel in critical positions maintain proof of suitability and trustworthiness. This list should be restructured to address the specific needs of the individual organizations.

CATEGORY: Computer virus scan/configuration check

SERVERS & PCs REQUIRING PERIODIC SCANNING

No. assigned	User/responsible employee name	Equipment identification (inventory no.)	IP address or host ID	Building ID	Room or area	Port or circuit	Date of last virus scan and update
VIR-001							

INSTRUCTIONS: Create and maintain an accurate list of the various PCs and servers that are part of the SCADA system or that share a LAN connection (even through a firewall) with the SCADA system, for the purpose of scheduling and verification of completion of periodic virus scanning and configuration checks on those computers. It is important to verify that all such computers have suitable virus-scanning software and that it is enabled for the specified checks. It is also important to verify that specified configuration settings (including digital certificates, disabling of Wi-Fi or Bluetooth, etc.) are properly established. In many organizations, the ability to have automatic virus-scanning updates (on computers within the electronic security perimeter) will not be possible, owing to firewall settings; therefore, someone will need to be responsible for periodically updating the virus profiles of these computers. If VPN technologies are used for remote access, it will be necessary to ensure periodic updating of digital certificates. The purpose of this list is to aid in organizing and scheduling these activities and in making sure that all relevant computers are identified. This list should be reviewed quarterly, whenever new equipment is added or existing equipment is removed, or whenever software upgrades (e.g., updating the operating system version) are installed.

CATEGORY: Electronic access points

LOCAL AREA NETWORK CONNECTIONS

No. assigned	VLAN segment description/usage	Protective measures implemented	LAN equipment ID	Building ID	Room or area	Port or circuit
LAN-001						

INSTRUCTIONS: Create and maintain an accurate list of actual communication connections (LAN) between the SCADA system and any other systems or equipment outside the defined electronic perimeter. This could include a LAN connection to the Engineering Department or to a GIS system or other such connections. If desired, this list can also be used to document the LAN connections of the actual SCADA equipment to the SCADA LAN, if no other document already exists for that purpose. The purpose of this list is primarily to document LAN-based electronic access points through which an external attacker could attempt to gain access to the SCADA system. The actual connection points (in the form of switch equipment, hubs, and ports used) should be defined to the level necessary to enable an authorized person to physically disconnect (or reconnect) and isolate the SCADA LAN from external systems. This information will also be useful in the configuration of a NIDS that monitors for unexpected traffic on the SCADA LAN and for ensuring proper configuration of Ethernet switches. Another key component of this list is the documentation of protective measures (or verification of the existence of suitable protective measures) being applied to secure this access point. The structure and content of this list should be modified, as needed, to incorporate sufficient information to permit management review of the protective measures in place. This does not have to include the details concerning the programming of rules in a firewall, but it should indicate that such provisions are in place, identify the document where those details are recorded, and document that the provisions are reviewed on a regular basis. With firewalls, there is a need for regular updating of virus profiles, and some mechanism needs to be in place to ensure that such updates are being made; that need not be part of this list, but this list should indicate the person responsible. This list should be reviewed on a biannual basis or whenever the equipment or network configuration of the SCADA system is modified. This list should be considered as a critical information asset and handled with the requisite security and confidentiality.

WIRELESS ACCESS POINTS

No. assigned	WLAN segment description/usage	Protective measures implemented	AP equipment ID	Building ID	Room or area	SSID
WAP-001						

INSTRUCTIONS: Create and maintain an accurate list of any Wi-Fi access points (APs) implemented on the SCADA LAN. The purpose of this list is primarily to document wireless electronic APs through which an external attacker could attempt to gain access to the SCADA system. The actual AP information should be defined to the level necessary to enable an authorized person to physically disconnect (or reconnect) and isolate the SCADA LAN from wireless access. This information will also be useful in the configuration of a NIDS that monitors for unexpected traffic on the SCADA LAN. Another key component of this list is the documentation of the protective measures (or verification of the existence of suitable protective measures) being applied to secure this AP. The structure and content of this list should be modified, as needed, to incorporate sufficient information to permit management review of the protective measures in place. This does not have to include the details concerning the disabling of beacons, enabling of WEP, and so on, but it should indicate that such provisions are in place, identify the document where those details are recorded, and document that the provisions are reviewed on a regular basis. This list should indicate the person responsible for maintaining the security configuration of the APs. This same list can also be used to track any PCs that are constantly or periodically connected to the SCADA LAN and that have integral Wi-Fi and/or Bluetooth capabilities. The user(s) of these computer(s) should be identified, and there should be a record

indicating that suitable configuration settings have been implemented on these PCs. This list should be reviewed on a biannual basis or whenever the equipment or network configuration of the SCADA system is modified. This list should be considered as a critical information asset and handled with the requisite security and confidentiality.

WIDE AREA NETWORK CONNECTIONS

No. assigned	WAN description/ purpose	Router/FRAD serial no./model	Protective measures implemented	Circuit ID number/type	Building ID	Room or area
WAN-001						

INSTRUCTIONS: Create and maintain an accurate list of actual communication connections (WAN) between the SCADA system and any other systems or equipment outside the defined electronic perimeter. This could include a WAN connection to the regional EMS or ISO SCADA system or other such connections. The purpose of this list is primarily to document WAN-based electronic access points (APs) through which an external attacker could attempt to gain access to the SCADA system. This list should include connections to the Internet via an ISP, leased telephone lines used for point-to-point network connections, frame-relay circuits, X.25 circuits, and T1/T3 circuits. The actual connection points (in the form of communications equipment and the ports used) should be defined to the level necessary to enable an authorized person to physically disconnect (or reconnect) and isolate the SCADA system from external systems. This information will also be useful in the configuration of a NIDS that monitors for unexpected traffic on the SCADA LAN. Another key component of this list is the documentation of the protective measures (or verification of the existence of suitable protective measures) being applied to secure this AP. The structure and content of this list should be modified, as needed, to incorporate sufficient information to permit management review of the protective measures in place. This does not have to include the details concerning the programming of rules in a firewall, but it should indicate that such provisions are in place, identify the document where those details are recorded, and document that the provisions are reviewed on a regular basis. With firewalls, there is a need for regular updating of virus profiles, and some mechanism needs to be in place to ensure that such updates are performed. That need not be part of this list, but this list should indicate the person responsible. This list should be reviewed on a biannual basis or whenever the equipment or network configuration of the SCADA system is modified. This list should be considered as a critical information asset and handled with the requisite security and confidentiality.

TELEPHONE CONNECTIONS (MODEMS)

No. assigned	Description/purpose	Telephone no.	Telecom circuit ID	Modem ID	Protective measures implemented	Building ID	Room or area
TEL-001							

INSTRUCTIONS: Create and maintain an accurate list of authorized telephone circuits that provide dial-out and/or dial-in connectivity to the SCADA system, as well as leased lines used for remote, dedicated, serial communications to other systems. This is not specifically a list of the leased telephone circuits used for RTU polling, although it may be used for that purpose if that information is not documented elsewhere. This list should not include any leased telephone line used to create a WAN linkage to another site/facility (e.g., a field site), because that would be a WAN connection (and thus should be listed in the previous table entitled "Wide Area Network Connections"). This list is intended to document primarily dial-in/dial-out telephone circuits that could enable an external attacker to communicate with the SCADA system, one of the computers that compose the SCADA system, or to be routed onto the SCADA LAN. Of primary concern are any such telephone lines that are permanently connected to a modem set up for auto-answering of an incoming call. Making and maintaining an accurate inventory of all such telephone circuits is an essential component of cybersecurity. The list should document the measures (dial-back, encrypting modems, VPN, manual connection of modem, etc.) used to secure this connection. Management should review the list and make decisions about the need for each, as well as about the suitability of the protection used to secure each. This list should be coordinated with accounting information related to any conventional telephone circuits (circuits not derived from a bulk T1/T3 circuit connected to the company's PBX), so that all such independent circuits are identified. If any circuit has multiple extensions, each should be identified. The employees authorized to use each circuit should be identified, as should the manager responsible for authorizing the usage. This list should be reviewed on a biannual basis or whenever the equipment or communications configuration of the SCADA system is modified. This list should be considered as a critical information asset and handled with the requisite security and confidentiality.

CATEGORY: Physical access monitoring & control

ENTRY POINTS INTO SENSITIVE AREAS

No. assigned	Area/section	Building ID	Floor/room	Entrance/ egress	Type	Access level	Control mechanisms	Last tested on	Alarms
ACC-001									

INSTRUCTIONS: Create and maintain an accurate list of authorized physical entrances into facilities and areas containing critical systems and information. All nonauthorized entrances (windows, loading docks, suspended ceilings, etc.) are assumed to be monitored and/or alarmed or sealed in an acceptable manner. The purpose of this list is to account for all allowable means of entrance and/or egress (e.g., a fire door) and to enumerate the control (and monitoring) mechanisms that will be used to secure these access points. Not all areas require the same level of access control or monitoring. This will be a decision that is made by the organization. This list is intended as an aid in formulating a strategy for access control and monitoring. Access controls may range from simple keyed locks to snares with biometric authentication, depending on the critical nature and contents of the area being protected. This will be up to the management of the particular organization. If monitoring is performed, the mechanisms can range from a simple log book to full-time video surveillance or even an armed guard. Again, the appropriate

mechanisms must be determined by the respective organizations. If entry requires that special authorizations be given and possibly the use of a reserved item (e.g., a special key), then this should be indicated in the list above. There should be written procedures for the testing of security mechanisms and for the upgrading and replacement of existing mechanisms. The mechanisms used for access control and monitoring should be reviewed and tested on a periodic basis, and this information should be added to the table. This list should be reviewed at least annually and whenever a change is made to the facility or an access security mechanism is replaced or serviced. This list should be considered as a critical information asset and handled with the requisite security and confidentiality.

Simplified checklist for self-assessment

Although collecting and organizing the information outlined in the various tables given in the previous section is an important part of organizing and planning an overall cybersecurity program, there are some simple questions that can be asked/answered that help you to understand how well (or poorly) your current situation meets overall cybersecurity requirements. If you can answer "yes" to the majority of the following items/questions, then you are probably in pretty good shape. If not, then you need to consider the consequences of your vulnerabilities and possibly develop a plan and program to reduce them to acceptable levels.

No.	Description/action/requirement	YES	NO	DON'T KNOW
PERSONNEL				
	Do all personnel sign employment agreements that include nondisclosure provisions?			
	Do all personnel (or at least those granted physical/electronic access to key systems) undergo a security check prior to employment and on a periodic basis thereafter?			
	Do all personnel receive basic familiarization with company security policies?			
	Do all personnel receive basic training in cybersecurity issues and social engineering?			
	Do key personnel receive more detailed training in cybersecurity technologies?			
	Is there an active security awareness program at the company?			
	Are there high-level managers assigned to be responsible for cybersecurity?			
	Are there compensation/incentive programs linked to security goals?			
	Are personnel permitted to bring their own software and computer equipment into the office?			
	Are personnel required to change their password(s) at least every 90 days?			
	Are fire drills regularly scheduled, and do all employees know their responsibilities?			

	Are employees trained in the handling of confidential information?			
	Are employees trained in the reporting of suspected security incidents?			
	Is senior management aware of the legal liabilities associated with improper information disclosure and with the due diligence obligations associated with SOX and HIPPA?			
	Are backup copies occasionally reloaded in order to verify their validity and to keep personnel trained on the necessary procedures?			
	Are there written procedures for critical activities, and are they practiced and reviewed (and updated) on a periodic basis?			

COMMUNICATIONS

	Have all communication interfaces into the critical systems and the facility been identified, including LAN, WAN, wireless, Internet, and telephone?			
	Do all WAN connection points have a suitable router with firewall and link encryption capabilities (and are they enabled)?			
	Is there a suitably configured firewall between the LANs of critical systems and any other LAN connection used to interconnect critical and noncritical systems?			
	Is there a person assigned to ensure that all firewalls are kept up to date with current security profiles and patches?			
	If Wi-Fi is utilized, are the APs configured with the recommended security features, such as 128 bit encryption and SSID suppression?			
	If Wi-Fi is used, does the facility employ any software for detecting rogue APs?			

PHYSICAL SECURITY

	Are all access ways into secure areas continuously monitored in some manner?			
	Are all exterior doors secured from access (although possibly not egress)?			
	Are all windows into secure areas protected with bars, security glass, or some other item?			
	Are all fire doors set up with alarms?			
	Does the fire alarm sprinkler system have a delay on release of water for manual alarm pulls?			
	Does the fire sprinkler system have separate, independently activated zones?			
	Are employee credentials of a form that is very difficult to forge?			

	Does unattended access control include multiple factors (e.g., ID card and PIN number)?			
	Is there a differentiation in the access credentials for different areas and categories of personnel? (Are access credentials universal, or are they area/employee-specific?)			
	Is attended video surveillance in effect for the most critical areas?			
	Is the recorded video routinely collected and stored in a secure manner?			
	Are visitor logs verified by a guard (or an authorized employee) and collected and stored on a routine basis?			
	Are electrical supply systems and HVAC controls for critical systems and areas protected from tampering?			
	Are entry/exit logs created by the access control system, and are they reviewed and preserved on a regular basis?			
	Are fire-proof storage cabinets available for on-site storage of critical information and backup?			
	Does your business liability and fire insurance provide adequate coverage for the loss and restoration of critical systems, business interruption, and potential legal liabilities if confidential information is improperly disclosed as a result of a cyberattack?			
	Are backups made on all computer systems on a regular (daily to weekly) basis, and are multiple copies on a storage rotation scheme with off-site storage?			
	Are there procedures to ensure that backup copies cannot be claimed by unauthorized personnel?			
	Is there a suitable UPS in place for critical systems (including habitability considerations and communications subsystems)?			

ELECTRONIC SECURITY

	Does the company have a policy prohibiting Wi-Fi access points on the LANs of critical systems?			
	Does the company have a policy regarding removable media?			
	Does the company have an email server that scans and sequesters attachments and blocks identified SPAM sites?			
	Does the company have a policy regarding use of the Internet, IM, and email for non-business purposes?			
	Does the company require that only approved, properly licensed software be loaded onto personal computers?			
	Does the company prohibit the use of personal computer and peripheral equipment or their attachment to company systems and networks?			
	Is there a NIDS monitoring traffic on the critical LAN/WAN segments?			

	Is there a HIDS monitoring the resources and applications on the set of critical hosts and servers?			

SYSTEM MAINTENANCE

	Are all computers equipped with the latest virus-scanning and firewall software and configured to receive automatic updates?			
	Is there a department/person responsible for ensuring that relevant security updates and patches are installed on all systems on a regular basis?			
	Do your system vendors perform functional testing of operating system, networking, and third-party software updates and patches and provide integrated updates that incorporate all of these items?			
	Do your critical vendors have a 24/7 support capability that includes trained, competent help desk personnel who are familiar with your specific systems?			

DISASTER RECOVERY

	Does the company have a written disaster recovery plan?			
	Is this plan reviewed, tested, practiced, and amended on an annual basis?			
	Does the company have an alternative site of some form, which can be put into operation if the primary site or systems are unavailable?			
	Does the disaster-recovery planning include and address telecommunications requirements?			
	Does the disaster-recovery plan address obtainment of necessary equipment and software resources?			
	Does the disaster-recovery plan address the return to the primary place of business and termination/disposition of the alternative site?			
	Is there a senior manager with responsibility for assuring the viability of the disaster-recovery plan and procedures?			

Glossary

A2D: Analog-to-digital conversion. The electronic circuitry that generates a binary numeric value that varies in value over a predefined range, directly in relation to an input voltage signal being varied over a corresponding voltage range.

ACL: Access-control list. In physical security, this is the list of personnel who may enter given secured areas, possibly with the added specifics such as days of week and times permitted and if they can escort others. This may be configuration data for an electronic access control system. For firewalls, this term refers to the rule set configured to define message traffic that may or may not pass through a firewall.

Address spoofing: Using hacking tools to construct Ethernet frames and/or IP datagrams that have falsified source address information in order to get past protective and detective cybersecurity mechanisms. IP spoofing often involves setting an internal IP address as the source address of an IP datagram sent from the outside.

AGA: American Gas Association.

AGA calculations: A series of ever more precise volumetric (or caloric) calculations for natural gas, based on the available instrumentation and analytical measurements.

Aggregator: A computer-based appliance (special purpose Ethernet switch) used to collect multiple Ethernet frame streams from various sources (e.g., SPAN ports from several other switches) and combine them, in real time, into a single combined stream. The "speed" (Mbps) of the inputs to the aggregator are typically 10/100 Mbps, whereas the output of the aggregator tends to be fiber-optic Ethernet with at least a 10 Gbps data rate so as to avoid loss of messages.

Alarm acknowledgment: A specific action (usually a keyboard function or operator display target—mouse click) taken to indicate to a SCADA system that newly detected and specially annunciated alarm indications have been reviewed, so that special annunciation (visual and audible) can be terminated.

Alarm filtering: A user-configurable and alterable mechanism for selecting only a subset of all current (or prior) alarms for presentation to the SCADA system operator.

Anomaly detection: A form of detection that looks for behavior or activity that is sufficiently outside the normally observed or baseline activities and behaviors as its detection methodology. Anomaly detection does not rely on pre-known

'signatures' but may require a pre-operational training/learning interval to develop a baseline.

ANSI: American National Standards Institute.

AP: (Wireless) access point. A Wi-Fi (wireless Ethernet) transceiver/gateway connected to a conventional Ethernet LAN, providing wireless, broadband bridging of the LAN to client devices within range that have a compatible wireless card.

API: Application program interface. The explicitly defined set of subroutines or library functions (or object methods) offered to an application program(mer) to access capabilities of the operating system, other applications, or system resources.

ARP: Address Resolution Protocol. A broadcast message type for Ethernet and other physical LANs that is used to obtain an Ethernet address when only an IP address is known (RAPR does the reverse—provides IP address when only Ethernet MAN is known).

ARP cache: The locally stored list of Ethernet MAC (and corresponding IP) addresses for all/many of the other computers on a LAN. Each computer (and also devices like a firewall) on a LAN maintains its own such list, which is updated by special ARP broadcast messages.

ARP cache poisoning: An attack wherein false ARP response broadcasts are sent onto a LAN in order to fill every device's ARP cache table with falsified information (sets every IP address in the table to have the MAC address of the attacker's computer).

Asymmetric: Unbalanced/unequal, usually in reference to the subdivision of communication bandwidth into a full-duplex channel (e.g., an asymmetric DSL circuit has much greater incoming bandwidth than outgoing bandwidth).

Asynchronous protocol: See character-oriented protocol.

ATM: Asynchronous transfer mode. (See also *SONET.*) A high-bandwidth WAN technology designed to provide high throughput as well as a high degree of determinism.

Authentication: Exchanging and validating credentials (often issued through a trusted third party) to prove an identity. In computer terms, this means exchanging and verifying X.509 digital certificates and possibly validating them with a CA elsewhere on the network.

Automatic/manual mode: Placing a periodically executed control calculation, used to manipulate a physical process parameter and control equipment (as a valve), under the manual control of a human operator or allowing the computer to continue the automatic execution of the calculation and logic.

B2B: Business to business. An automated exchange of data between the business systems of company A and company B, often using XML technologies (e.g., automated ordering from a vendor on the basis of inventory levels).

Backdoor: An undocumented, unauthorized, secret mechanism for gaining access to a computer system, such as a hard-coded ID/password or a Trojan horse program that relays information around the protective mechanisms of the operating system. Backdoors typically allow system access without going through the legitimate user login authentication process.

Bacteria: A self-replicating type of malware that multiplies and consumes all available memory, CPU, and disk resources until the operating system is disabled.

Baseline: The minimum set of (obligatory) technical requirements. Add-ons are available to address variable and differing needs. In regards to software, this is the basic set of applications, utilities and services used to set up a new computer.

Battery backup: A power source that can non-disruptively continue to supply DC power to operate computer equipment or protect volatile storage in the event of the loss of normal power sources.

Biometric scanner: Any device that makes one or more physical measurements of features unique to an individual (e.g., fingerprints, retinal pattern, and/or hand geometry), for confirmation of identity.

Bit-oriented protocol: A legacy serial protocol that typically predates the development of UART technology and microprocessor-based RTUs, using asynchronous messages with frame lengths that are not definable as an exact multiple of eight-bit octets but rather some specific fixed-bit length (e.g., 125 or 247 or 518 bits per message frame) defined in the protocol specification.

Blacklist: A list of items (e.g., IP addresses, URLs, Web sites, applications, etc.) that are prohibited from access. Often implemented as part of a personal firewall or HIDS. The opposite approach from the technique called 'whitelisting'.

Bluetooth: A wireless LAN technology for short-range, ad hoc communications connectivity between computer devices and peripherals and for use in cell phones and wireless speakers and headsets.

Boolean logic: Predicate logic equations based on the two-state (true/false) combinational algebra defined by George Boole in the 1800s. Inputs are acted on by the operators AND, OR, NOT, NOR and XOR to produce a true/false result.

Buffer overflow: A type of hacker attack in which programs that accept data input are given more data than they expect. The excess data (which is actually carefully crafted computer instructions) target the program logic in the memory region that follows the end of the memory storage area allocated to hold the data input, overwriting legitimate computer instructions and replacing them with instructions selected by the hacker.

CA: Certificate authority. A third party that provides (sells) X.509 digital certificates for authentication purposes, also providing online authentication (or rejection) of certificates that are alleged to have been issued by that CA. The CA acts as an impartial third party that can verify that the credentials offered for identification to computer A by computer B are valid and were issued by the CA to computer B.

Calculated variable: A numeric or Boolean value that is computed by the computer system, from other physical inputs, manually entered inputs, or other calculated variables, and that is subsequently treated by the SCADA system as if it were an actual physical input.

Challenge-response: A class of user authentication that provides the user with a query that has a pre-defined response that should only be known to the legitimate user. For example, many financial institutions require you to provide answers to several questions (e.g., name of your grade school) that an attacker should not know, and then they will subsequently use those questions and your response to authenticate you.

Character-oriented protocol: A protocol that uses messages composed of one or more ASCII characters or eight-bit octets, thus making the messages suitable for use over asynchronous serial communication circuits on the basis of conventional computer serial-port UART hardware designs.

Check-before-operate: A supervisory control mechanism and associated RTU protocol that requires a three-step process, with operator involvement, in order to affect a control operation with a selected RTU. A.k.a., select-check-operate.

Cipher: A scheme for encrypting messages (plaintext) into a nonreadable form (ciphertext) and then back into plaintext.

Ciphertext: The encrypted form of a message once it has been subjected to a cipher or encryption algorithm.

Circular buffer: A fixed-length (e.g., *n* elements) data storage area where, once filled, each subsequent incoming data element overwrites the oldest data previously stored into the buffer area, so that the buffer always holds the most recent *n* elements.

CLI: Command-line interpreter. An operating system utility program that interacts with a human user to allow the entry of commands, from a predefined set, in human-friendly form, to be executed by the operating system. The Microsoft DOS window (cmd.exe) is an example of a CLI.

Client: A computer application that makes a communication connection to another application in order to request that application to provide some form of service that the client requires (e.g., a Web browser client and Web server). Half of a client-server pair of applications.

Closed-loop control: A computer-based control algorithm. The computer makes a measurement, computes the need for (and magnitude of) a control adjustment, and automatically sends the necessary commands to the respective control device, to make an adjustment without the need for human intervention.

Collision: A term used in reference to non-switched Ethernet LANs when two or more nodes attempt to initiate transmission of an Ethernet frame concurrently and thus interfere with each other's transmission.

COM: Common/Component Object Model. The Microsoft equivalent of CORBA. COM was intended for applications running on the same computer and was extended by DCOM (distributed COM) which allowed for applications on different computes, but on the same LAN. COM messages allow application programs to share data and provide processing services to each other.

Command line interpreter: The basic textual user interface, provided by a computer-based device, which allows a user to execute a range of pre-defined commands in order to configure, test and manipulate the device. Similar to the cmd.exe "DOS" application provided in Windows. Common in devices such as firewalls, switches, and routers, but also smart instruments and analyzers.

Community string: The read and write passwords (text strings) sent from an SNMP client to an SNMP server as part of every message and used to authenticate the message sender to the server. Universal factory default values for these strings are "Public" and "Private." Only SNMP v3 protects these passwords with encryption.

Compiler: A program that converts human-readable commands (high-level programming language) into the actual computer command code numbers (machine language) used by a specific type of computer. This is different from an Interpreter which is a program that accepts a high-level, machine-independent program as its data and performs the functions called for in that data as if it were a real set of machine instructions for that specific computer.

Concentrator RTU: An RTU that acts as a master to one or more slave RTUs for the purpose of collecting their input data and integrating it into its own database so that a SCADA system needs only to communicate to the concentrator RTU and not also to the various slave RTUs.

Console port: The interface, in most operating systems, through which an authorized user is given direct access to the CLI (as opposed to a port that is owned by an application program through which a user would interact only with that application program). In older computers, this might be a physical port, but today it is more likely a logical port (e.g., the DOS window that can be displayed under the Microsoft Windows operating systems).

Console server: See terminal server.

Control-loop cascade: In complex, multivariable regulatory control schemes, there may be two or more control loops, the output of one (the upper loop) being the set point of the next (the lower loop). The data connection that passes the value of the set point from the upper loop to the lower loop is the cascade.

CORBA: Common Object Request Broker Architecture. In Unix, the technology for allowing objects to be shared by applications running on different computers, even on different operating systems.

CPU: Central processing unit.

Cracker: A hacker who uses computing/programming knowledge and skills to cause damage and harm to, or gain the unauthorized use of, the computer systems and networks of others.

CRC: Cyclic redundancy check. A binary polynomial calculation used to generate a unique binary number for a unique sequence of binary data provided as input.

CRC code: Code number carried along with transmitted messages to verify the message integrity at the receiving end. A form of one-way hash algorithm used in most modern communication protocols.

CSU/DSU: Channel service unit/digital service unit. The electronics and logic needed to interface with both the communications equipment and the end digital device requiring connectivity. FRAD is an example of a CSU/DSU device, as is an ISDN modem. A CSU/DSU is commonly used to connect to a T1 telephone circuit.

CSV: Comma-separated value.

CSV file: An ASCII text file in which data values are written as ASCII character sequences with a comma character separating each subsequent data element. Commonly used for data exchange between Microsoft applications, such as Excel spreadsheets.

Daemon dialer: A hacker tool (program) for detecting modems. A daemon dialer will sequentially automatically dial all telephone numbers within a defined range or pattern, recording the numbers answered by a modem. Some daemon dialers do additional probing when a modem answers, to glean additional information for the hacker.

Database point: In a real-time control system, each piece of information that is part of the real-time database, updated on a regular basis either by RTU polling or via calculations. A database point (also called a tag) is usually a specifically named data structure that includes a current value that can be used in displays and reports and tested for alarm limit violations. A point (or tag) is usually a collection of data elements, in a defined structure, including the actual (most recently updated) value of the point. The other data elements describe the point and how it is to be displayed and processed by the control system.

Datagram: A self-contained data packet carrying its own address information, so it can be independently routed from the source to the destination computer without reliance on earlier exchanges (between the source and destination computer and the transporting network). In TCP/IP architecture, IP datagrams (or fragments thereof) are the basic messages sent through an IP-based network, carrying UDP or TCP packets as their data.

Dark web: Web sites on the Internet that do not use conventional HTTP/HTTPS protocol and thus cannot be viewed with conventional Web browsers. The Tor browser is a free and popular browser that can fetch and render ".onion" files (dark web pages). Dark web sites are often used for drug sales, sales of stolen personal/financial information, and for selling 0day exploit code.

Data mining: The process of looking for previously unrecognized patterns or statistical relationships among huge data sets by use of statistical, neural network, or other means.

Day-zero virus: A new virus that has just been launched onto the Internet and thus is not known to and may not be recognized by the current virus-detection software. Heuristic virus-detection programs attempt to identify such viruses by similarities to known viruses; IDS packages attempt to identify such viruses by their actions once they enter the IDS-monitored system.

DCOM: Distributed Common Object Model. The extension of the Microsoft COM architecture for applications running on network-connected computers, as well as applications running on the same computer.

DCS: Distributed control system. An automatic process control system composed of multiple, independent control computers, each responsible for only part of the overall plant/process. (In theory, a partial DCS failure will not take down the entire plant/process.)

DDoS: Distributed denial of service. (See DoS.)

Dead-man timer: Hardware and/or software that monitors operator activity and initiates some action if no activity is detected within a predefined time interval.

Deadband: A value range around (plus and minus) a given measurement alarm limit or setpoint within which the value can vary without setting off an alarm indication. Usually expressed in absolute measurement amount or in a percentage of measurement scale.

Default gateway: The device on a LAN segment that provides external connectivity and to which messages will be sent for forwarding to their final destination when the sending computer/device determines that the destination IP address must not be on the local network. A default gateway can be a router, firewall, or even a doubly connected computer.

Default permit/deny: A firewall configuration setting that is applied to packets attempting to transit the firewall when none of the defined ACL rules could be applied.

Denial of service: See DoS.

Decryption: The process of undoing the encryption of a ciphertext message to recover the original plaintext message.

Deep inspection firewall: A firewall that looks at the message content of IP packets and is knowledgeable of defined protocols and malware, to verify that the IP packets do not contain a known virus and conform to the protocol specifications. (See also *stateful inspection* and *application proxy.*)

Demand scan: A SCADA supervisory function whereby an operator can force the immediate polling of one or more RTUs, thus preempting the scheduled RTU polling activity.

DHCP: Dynamic Host Configuration Protocol. A protocol in the IP suite that is used to incorporate a computer into an IP network by assigning it a temporary (reusable/dynamic) IP address from an available range of addresses.

Dictionary attack: Attempting to break (guess) the password of a target computer system by going through a previously compiled list of words (the dictionary) and trying them in succession. The word list is usually oriented around a given theme, such as sports teams or city names.

Digital certificate: A set of data that can be used as electronic credentials to verify a computer's identity by another computer. A digital certificate typically contains identification information, a digital signature's validity dates, and a unique serial number, as well as the sender's public encryption key and information about the issuing CA (including their digital signature). The ISO specifies a standard for digital certificate structure and content, currently X.509 v3.

Digital signature: An encrypted data element that cannot be forged (using reasonable means) and that uniquely verifies the sender of a message and confirms that the message has not been altered. Usually included as part of a digital certificate.

Digital-to-analog conversion. An electronic circuit that produces a voltage output that corresponds to the binary numeric value input to the circuit. Occasionally abbreviated as A2D.

Digitization: Taking sequential samples of an analog signal and converting them to a sequence of numbers, with an A2D converter, such that the numeric sequence adequately reproduces the analog signal when sent back through a digital-to-analog converter circuit at the same sampling rate. For digitization and transmission of the human voice/speech, the telephone company uses 8,000 samples per second (stored as eight-bit integers).

Display drill down: Going from a high-level overview display to a more focused and detailed display by using poke points and navigational targets built into an HMI application.

Display hierarchy: Logical structuring of navigational relationships among a large number of available display pages, facilitating the location of the desired display page in an intuitive manner by use of the fewest possible intermediate steps.

Display navigation: Switching from display page to display page by using the navigational poke points and keyboard functions (e.g., soft keys or function keys) provided by the operational HMI.

DLCI: Data link connection identifier. The communications interconnection address/circuit address used to interface with a frame-relay network. An end point node on a frame-relay WAN is uniquely identified by its DLCI, much as an IP address identifies a computer connected to the Internet. The DLCI is included in all messages and is used for routing within the frame-relay WAN.

DMA: Direct memory access. Special hardware/circuitry that allows the writing of bulk data directly into a contiguous range of random-access memory (RAM), without passing through the CPU under program control. Commonly used for reading and writing blocks of data to/from bulk storage devices (e.g., hard drives).

DNP3.0: Standardized serial protocol for use with RTUs and other IEDs. Supports a reasonable range of strong data types, as well as polled and unsolicited report-by-exception modes. Widely used in the electric utility market. An IP version has also been defined.

DNS: Domain name server. The computer that provides URL-to-IP-address conversion, when requested by a Web browser or an email client/server. Like a telephone directory, but converting a URL, rather than a name, into an (IP) address.

DNS cache: The local storage of a limited number of recently accessed URLs and their corresponding IP addresses, eliminating the need to go to a DNS. A point of attack by hackers, who overwrite the IP addresses with different values.

Domain name: A uniquely assigned, registered name (e.g., mywebsite.com) that is added to the applicable DNS servers, so that URL references used in emails, for example, can be converted into an actual IP address.

Domain name server: A computer that is running a DNS service and thus able to provide a lookup service to provide an IP address when given a URL containing a domain name.

DoS: Denial of service. An attack in which communications with a computer or device are blocked, typically by overloading the bandwidth of the communication channel and/or the computing power of the computer/device. A common mechanism is to have a large number of zombie computers concurrently deluge the designated computer/device with a constant stream of messages; this is called a distributed DoS (DDoS) attack.

Drill down: See display drill down.

DS0 circuit: The lowest-bandwidth digital communication circuit offered by the telephone company. Offers the bandwidth defined by the telephone company as needed to deliver the digitized human voice (i.e., 8,000 samples per second × 8 bits per sample = 64 kbps).

DSL: Digital subscriber line.

DSSS: Direct-sequence spread spectrum. A type of spread-spectrum radio that uses a pseudo-random chipping code, combined with the actual data being transmitted, to make the resultant signal appear as random noise to any receiver that can't undo the effect of the chipping.

Dual ported: An RTU that has two (or more) serial ports that can be independently used by separate SCADA systems to poll and retrieve data. Commonly used in the electric utility industry. Dual ported RTUs are occasionally used for the purpose of protocol conversion where different protocols are used on the ports, and the RTU has no physical I/O hardware.

Dynamic IP Address: An IP address (and associated information) provided to a computer or device from a DHCP server using the DHCP protocol at the reboot or restart of the computer/device. It is called dynamic partially because of the possibility of the same computer/device receiving different IP addresses.

EGU conversion: The mathematical computation of a process engineering unit value *Y* (e.g., volts, cubic feet per minute, degrees Fahrenheit, etc.) from a binary integer *X* returned by an A2D converter circuit. Often (but not always) a linear relationship in the form of $Y = MX + B$.

Electronic vaulting: Sending critical data from one computer over a network to another computer, where it can be stored as a backup in the event that the first computer is damaged or the data it contains are lost.

EMI: Electromagnetic interference. A category of electronic noise that can alter (introduce errors into) a message transmitted over a communication circuit. (Additional usage: entity-machine interface—for the extremely politically correct, who don't wish to risk offending nonhumans by using the term HMI.)

Encapsulation: Transporting one type of message (e.g., individual ASCII characters or serial protocol messages) as the data portion of another type of message. Specifically, using TCP/IP messages to carry non-IP messages across an IP network.

Encryption: Manipulating a message, in a reversible manner, to prevent unauthorized personnel from being able to read the message by making the message incomprehensible (see *cipher*).

Encryption key: See *key.*

Engineering units: Numeric values that represent physical measurements in the actual physics/chemical terms used for the measurement (e.g., DegC, PSIG, GPM, etc.), rather than in a binary numeric value that corresponds to the actual measurement.

Ethernet: A LAN technology used for high-speed computer interconnections. Available in bandwidths of up to 10 Gbps and in wired, fiber-optic, and wireless (Wi-Fi) forms.

EtherType: A hexadecimal value greater than 0x5DC (hex) places in the length field of an Ethernet frame used to indicate the contents of the frame's data area.

Ethical hacker: A computer expert who uses computing/programming talents and skills to identify, report, and correct cyber vulnerabilities, rather than to exploit them (also called a white-hat hacker).

Evil-twin AP: A nonauthorized wireless access point (AP) placed in physical proximity to the real AP, broadcasting a replicated (cloned) identification, so that users log onto the evil-twin AP, rather than the valid AP. The attacker can record packets to get (and spoof) MAC address and WEP authentication-key information.

Extranet: A stand-alone, totally independent, IP-based network that is not interconnected to the Internet.

False negative: Deciding that a person's biometric readings are not sufficient to match that person's recorded readings, thus denying access when this *is* the person in question (i.e., denying access improperly).

False positive: Deciding that a person's biometric readings are sufficient to match the recorded readings of the alleged personnel, granting access when this *is not* the person in question (i.e., granting access improperly).

FDM: Frequency-division multiplexing. Sharing a single communication channel by breaking it into separate frequency bands or subchannels.

FHSS: Frequency-hopping spread spectrum. See *frequency hopping*.

Fieldbus: The generic term for a moderate-to-high-speed LAN technology used to interconnect a small to moderate number of smart instruments and control elements to enable them to interact as well as to provide communication access to these instruments and control elements from an IACS such as a PLC or DCS. Examples include Foundation Fieldbus H1 and Profibus. Similar to proprietary vendor LANs except published as open standards. Now they also include Ethernet-based versions.

Firewall: A computer-based security appliance placed between two networks for the purpose of vetting packets going in either direction, blocking those that do not meet predefined rules and guidelines. Firewalls range in capability from simple packet filters to deep inspection (and stateful inspection) versions and even application proxy servers that are protocol-aware.

Firmware: The application-specific software that is typically placed into an embedded, special-purpose computer/device and that is never (or rarely) modified or upgraded. Generally, this software is stored in nonvolatile memory, such as Flash or PROM (Programmable Read Only Memory).

Flash drive: A storage device that uses non-volatile flash memory, has no moving parts, and has a USB interface with a Type-A plug through which it is powered.

Flash memory: An electronic (solid-state) non-volatile computer memory storage medium that can be electrically erased and reprogrammed. The individual flash memory cells (bits) consist of floating-gate MOSFETs, which retain their state without power required.

FRAD: Frame-relay access device. Router-like networking appliance that makes the physical connection to the frame-relay WAN and provides serial ports and Ethernet LAN connections to the devices at the particular site. Also used to mean frame-relay assembler/disassembler, one of the internal functions of the FRAD.

Fragment: A portion of an IP datagram that has been created by breaking the datagram into smaller pieces in order to not exceed the maximum message size supported by the underlying physical network layers (e.g., Ethernet with a data size limit of 1500 octets).

Frame relay: A moderate- to high-bandwidth packet-switching WAN technology offered by commercial suppliers (usually the telephone companies) that is replacing older, X.25 packet networks.

Frequency hopping: A type of spread spectrum radio in which the transmitter makes channel changes, on a pseudo-random basis, during the transmission

of a message, to make monitoring difficult for a receiver not suitably equipped to follow the same pseudo-random channel sequence. This scheme enhances transmission security and allows for better sharing of the available channels among a large group of geographically adjacent users. The primary advantages of frequency hopping radios should be considered as avoiding interference and optimal channel usage, and not increased security.

FTP: File Transfer Protocol. An application-layer protocol in the IP suite. Used to send a copy of a file from one computer to another, over an IP network. It includes error checking and retry logic for transfer reliability.

Full-duplex channel: A communication channel that supports concurrent bidirectional data transmission.

Gateway: A communications appliance used to connect two different types of networks. A gateway deals with issues like the electrical/interface and protocol differences between the two networks.

Generator ramping: Sending control commands to RTUs that are located at generating facilities and interfacing the generating units. Typically, generator ramping provides a stream of momentary contact closures (pulses) to either increase or decrease the speed of the generator.

GMT: Greenwich Mean Time. A time standard based on the prime meridian (crossing through Greenwich, England) as the starting point for counting time differences between the time zones of the world.

GPS: Global Positioning System. A global network of special-purpose satellites that broadcast signals allowing suitably equipped receivers/computing devices on the surface to make a calculation of their exact latitude and longitude (and altitude), within a few meters of precision.

Graceful degradation: The architectural design of a computer system such that individual component failures will not totally shut down the functions of the system. A.k.a. fault tolerance.

Hacker: A computer expert who uses computing/programming skills to attempt to obtain unauthorized access to computer systems or to cause damage to those systems. (See also *cracker* and *ethical hacker.*)

Hacker tools: Assorted software utilities, developed by hackers and made available over the Internet, that enable users to launch attacks on computer systems. These tools often contain Trojans that infect the systems of those who download and attempt to use them. Many such tools can serve both beneficial purposes or malicious purposes based on who is using them (e.g., Pen tester versus malicious hacker).

Hacker Web sites: Where hackers share information about vulnerabilities in computer systems and methods for invading and attacking them—and post software utilities (hacker tools) that can be used for staging such attacks. Hacker Web sites can be found on both the "normal" Internet and on the so-called "dark web."

Half-duplex channel: A communication channel that supports message transmission in both directions, but not concurrently.

Hash code: An algorithm that takes a block of data and produces a unique numeric value, which cannot be used to regenerate the original data and which should never be the same for two differing blocks of data, no matter how slight the difference between the two data blocks.

HIDS: Host-based intrusion detection systems. (See also *NIDS* and *N/HIDS*.)

HIPPA: Health Insurance Portability Protection Act of 2004. A government regulation that includes provisions regarding the required security and protection of personal information related to health issues, in corporate computer systems.

HMI: Human-machine interface. The displays, keyboard, mouse, touch screen, and other equipment that together present information to and receive commands/data from a human being. (Formerly called *MMI*.)

Hot spot: A Wi-Fi access point (AP) available to the public (possibly for a fee) at such locations as hotels, airports, and shops. (Also called *Wi-Fi hot spot*.)

HTTP: Hypertext Transfer Protocol. A layered protocol in the IP suite that is used for messages between a Web server and a Web client (browser).

HTTP/S: A secure version of HTTP that sets up a session key and encrypts subsequent message traffic between the Web server and Web client (browser). Note that HTTP/S is vulnerable to a man-in-the-middle attack and does not authenticate the browser, only the Web server.

ICCP: Inter–Control Center Communications Protocol (also called UCA1.0 and TASE.2). A layered protocol in the IP suite that is designed for the exchange of many types of simple to complex information, and even commands, between SCADA systems. Primarily used in the electric power industries, but gaining popularity in water/wastewater applications.

ICMP: Internet Control Message Protocol. A set of messages used for management of the traffic across the Internet, including notifications of deleted packets and delivery failures and determining connectivity. Ping is an example of an ICMP message type.

IDS: Intrusion-detection system. Application software that runs in each computer on a monitored network. IDS monitors applications, resources, communications traffic, and other parameters that could indicate anomalous behavior caused by an intrusion. Variations of this are NIDS, HIDS, and N/HIDS.

IEC: International Electrotechnical Committee. An international standards association that deals with electronics and communications technologies.

IEC601131-3: A set of standards that define a series of programming conventions for PLC/DCS control logic, including conventional relay ladder logic, as well as sequential function charts, structured English, and other conventions.

IED: Intelligent electronic device. Electric industry term for microprocessor-based devices used in substations, including RTUs, protective relays, power meters, and fault recorders.

IIoT: Industrial Internet of Things. This refers to interconnected sensors, instruments, and other devices networked together with computer-based industrial applications to improve efficiency and optimization. The IIoT concept is an evolution of the distributed control system that theoretically allows for a higher degree of automation distribution.

Infinite loop: A program logic error that causes a program to continuously execute a set of instructions that contain no conditions/tests that would cause the program to reach an end point and to terminate.

Integrity scan: A polling request to an RTU requiring a full report of all inputs (and possibly outputs) and other data available within the RTU. Often used when a SCADA system starts up, to refresh its real-time database. Also used on a periodic basis when (unsolicited) report-by-exception polling is used, in case exception messages were missed or lost.

Interoperability: The ability of (different) computer operating systems and/or applications running on different computers to communicate and exchange information.

Interpreted language: A computer programming language that when/if "compiled," turns into data and not actual machine code instructions and thus requires another program (an interpreter) to read the data and perform the associated function. A.k.a. a script or scripting language.

Interpreter: A program that accepts a data file containing numeric codes defining the actions and operations supported by the program and to be performed in the sequence specified in the data file. The data file can be considered as a program that will be executed by the interpreter program. Scripting languages such as PHP and JavaScript are executed by interpreter programs built into the Web server or browser.

Interrupt: A software signal or electrical trigger used to change the value of the instruction pointer in a computer CPY in order to cause the computer to begin executing program code designed to service the condition that initiated the interrupt.

Intranet: An autonomous IP-based network that is interconnected with the actual Internet.

Intrusion detection: Identifying the presence of an unauthorized user, application, or malware, by monitoring of measurable operating system and networking parameters (e.g., message traffic, CPU usage, and file access). (See also *IDS.*)

I/O: Input/output.

I/O hardware: Electronic circuitry that provides an interface between a computer and physical process parameters and equipment, so that the computer can read process parameters and control plant equipment. Usually one of three categories: analog, discrete/digital, or pulse.

IP: Internet Protocol. The actual protocol used to send packets across the Internet.

IP address: The unique 32 bit (IPv4) or 128 bit (IPv6) numeric identifier that is assigned to any computer or computer device (as a router or gateway) that is directly connected to the Internet. IP addresses are registered with regional/national servers that track them and provide URL translation. IP addresses can be static (permanent) or dynamic (temporary).

IP_{SEC}: Internet Protocol security. Authentication and encryption functions embedded within the IP stack at the IP (network) layer, rather than as an add-on like SSL. It makes use of extended versions of the IP message packet that support the required authentication (AH), encryption (ESP), and key-exchange (IKE) functions and protocols.

IP spoofing: A hacker technique of generating IP message packets with a falsified sender IP address, to convince a packet-filtering firewall to allow the messages to pass because the sender appears to be a trusted entity.

IP to the field: The extension of the SCADA system (or corporate) IP-based broadband network out to field sites as a replacement for traditional serial, low-bandwidth RTU polling circuits. Requires either that field-based RTU/PLCs/system support an IP-based protocol or the use of a terminal server device in the field.

ISA: Instruments Systems and Automation (originally the Instrument Society of America but changed in the 1990s to keep up with changes in technology). An organization dedicated to process and plant automation and automated measurement and control.

ISDN: Integrated services digital network. An international communications standard for providing digital data and voice services on a common telecommunication circuit.

ISO: (1) International Standards Organization. The association that develops and issues standards such as the OSI seven-layer model for network architectures. (2) Independent System Operator. A regional organization that manages power markets and buys power from nonregulated electric power-generating companies.

ISO/OSI model: A deconstructed computer communications architecture broken into a stack of seven functional layers, each of which performs a specified set of functions and interfaces the adjacent layer(s) via well-defined interfaces.

ISP: Internet service provider. A company that provides a gateway to the Internet, for occasional and permanent users, through dial-up, DSL, satellite, and cable connections. ISPs also provide fixed IP addresses and temporary DHCP-based IP addresses.

Java: An interpretive programming language that is compiled into data called bytecode; bytecode can then be executed by a Java runtime engine (interpreter) such as are contained in many Web browsers and operating systems.

Java Applets: Small Java objects derived from a base class object that has been designed with many limitations (see sandbox) and sent to a Web browser as

part of Web page rendering, and used to add animation and data exchange to Web pages. Applets are a form of mobile code and potentially a cybersecurity threat, particularly as there are demonstrated ways to break the sandbox limitations.

IT: Information technology. The group that is responsible for specifying, purchasing, installing, maintaining, and upgrading the computing hardware and software assets of a company, including LANs, WANs, Web sites, portable computing platforms, and business software.

Key: The secret numeric value that is used in a cipher or encryption scheme, to convert plaintext into ciphertext. The encryption scheme can be made known as long as the key is kept secret. (Also called *encryption key*.)

KVM: Keyboard, Video, and Mouse—usually used in reference to some form of switch that connects those human-interface peripherals to one of two or more computers to allow the user to interact with that computer.

LAN: Local area network. Today, LANs are invariably based on Ethernet technology. (See also *PAN, VLAN,* and *WLAN.*)

LANman: The Microsoft LAN management software used with DOS and early Windows distributions to provide domain administration, and workgroup services.

Leak detection: For a SCADA system, specialized application software that uses real-time and totalized measurement data from RTUs in order to determine the likelihood and general location of a leak in a liquid or gas pipeline.

Learning/training period: For detection systems that use anomaly detection of some sort, there may be a need to allow the system to merely monitor activity for some period of time, while conditions are normal, in order to establish a baseline of some sort against which to compare activity once the system is put into detection mode.

M2M: Machine to machine. A type of plant/facility networking technology for eliminating sensor and I/O signal wiring in industrial monitoring and control applications.

MAC: (1) Media access control. The hardware and software rules that define how and when a computer may obtain the right to transmit a message onto a shared channel. Part of the layer of the ISO/OSI stack, just above the physical layer. (See also *MAC address.*) (2) Message authentication code. A hash-coded numeric value that is computed from and sent along with a message, to verify that the message has not been modified.

MAC address: The unique 48 bit Ethernet address programmed into every Ethernet controller board and Wi-Fi device. (Currently, computers/devices are migrating to a 64 bit replacement version called EUI-64.) Although MAC addressing is not specific to Ethernet, this is the most common usage. (See *MAC* [1].)

Machine code: The actual binary numeric codes that represent CPU instructions, which can be written into RAM and then executed by the computer. Assemblers convert assembly language programs into machine code.

MAN: Municipal area network. (See also *WAN*.)

Map board: A wall-filling physical depiction of a pipeline service area with graphical representations of the pipeline and field sites, which incorporates visual indicators and numeric readouts that indicate field conditions. A.k.a. mimic-panel.

Message payload: The data portion of a computer message. That is, the information being transported, separate from header and the routing and validation portions.

Metallic circuit: An electrical circuit connecting two or more stations/nodes that is constructed of actual wire or cable such that it provides an electrical circuit between/among those stations, e.g., an RS-485 multidrop 4-wire circuit.

MIB: A management information base (MIB) is a database used for remotely monitoring and managing the entities on an IP-based network. While intended to refer to the complete collection of management information available on an entity, it is often used to refer to a particular subset, more correctly referred to as MIB-module.

Microwave: The portion of the electromagnetic spectrum that sits between infrared radiation and radio waves, spanning 0.3–300 gigahertz (30 cm to 1 mm wavelength).

Middleware: Software used to link (i.e., provide interoperability) disparate applications, by performing the necessary data type conversions and supporting the messages of each connected application. Based on a set of distributed agents or on a central data repository.

Minicomputer: Typically a computer with a 12 to 16 bit CPU, a backplane with slots for peripheral interface boards, and RAM of between 4kB to 64kB, built in the late 1960s and 1970s.

MMI: Man-machine interface. See *HMI*.

Modem: Modulator/demodulator. An electronic device that converts binary digits (represented by specific voltage levels on an EIA-232 serial port) into audio tones suitable for transmission through the telephone system or over a voice-grade radio channel.

MP3: A compressed audio file (with encoding that sacrifices some of the audio quality in order to reduce the file size) corresponding to the published standard MPEG-1, audio layer 3.

MTU: Master terminal unit. Either the host computer in a SCADA system or a pre-SCADA electronic device that acted as a data concentrator for the RTUs.

Multi-dropped: A physical, electrical communication circuit that connects three or more nodes or stations and has the necessary electrical characteristics for circuit sharing.

Multi-factor authentication: Verifying a user's identity by requiring two or more sources of identification, such as a password and either an ID card or a biometric measurement or some other form of credential.

Multiplexing: The process of sharing a common resource (e.g., an A2D circuit) by placing some form of controllable or constantly cycling selection circuit between that resource and the items that need to use the resource.

NERC 1200/1300: The North American Electric Reliability Council's recommendations for improving the cyber-/overall security of the SCADA infrastructure for electric power generation and transmission.

NERC CIPs: NERC standards developed after, and replacing, 1200/1300 and which are still being enhanced, modified, and augmented as of 2020.

Network mapper: A utility that sends series of specially crafted IP messages (particularly ICMP messages with increasing TTL values) into a network, in order to identify the various computers through which the packets travel, and to probe the IP address range to see what computers/devices respond.

Network tap: A passive device that can be placed in a full-duplex Ethernet circuit for the purpose of replicating frames going in either direction to a "monitor" port, which can then be connected to a device such as a NIDS sensor, a VoIP recording device, or a computer performing sniffing. A.k.a. Enterprise tap.

NIDPS: (Network Intrusion Detection and Prevention System, a.k.a. NIPS) A form of NIDS where the sensors are placed in-line so that they can block malicious traffic rather than just monitoring such traffic.

NID: Network intrusion detection. Monitoring the communications traffic on a network, probably with a separate computer appliance on each LAN segment, to identify anomalous message traffic, IP addresses or protocols that could indicate the presence of an attacker attempting to penetrate the network, or malware sending messages across the network. A NID initially develops a characterization of normal network traffic by monitoring the traffic and then begins comparing observed traffic to the characterization, as a way of identifying anomalous traffic.

NIDS: Network-based intrusion-detection systems. (See also *HIDS* and *N/HIDS.*)

NIDS sensor: A computer appliance that is connected to a LAN (possibly using a switch SPAN port or a tap) for the purpose of examining every Ethernet frame passing that point in the LAN for malicious or suspicious content. A sensor may use signature and/or anomaly detection mechanisms and may pass data to a management console for more advanced analysis.

Offline storage: The transfer of bulk data and files to removable media that can be placed into secure storage. Recovery of the information requires that the particular removable media be located and installed into the drive attached to the computer. A scheme often used when the data archive interval desired (and the corresponding data quantity for that interval) greatly exceeds the practical online storage capacity of the computer system.

OLE: Object Linking and Embedding. A Microsoft application standard and protocol for distributed objects that allows an application to call on another application to process relevant parts of its data. (For example, a Microsoft Word document containing a spreadsheet would cause Word to call on Excel to process and render the spreadsheet portion of the Word document.) OLE replaced DDE and has itself been replaced by DCOM and ActiveX controls.

Online storage: Data stored on the bulk storage file system of a computer which can be rapidly accessed by applications programs with no need for manual intervention.

OPC: OLE for Process Control. A standardized mechanism for the exchange of data between programs that produce data (servers) and programs that wish to receive that data (client). Based on the Microsoft OLE architecture and DCOM technology. There are several classes of defined OPC servers, including servers that provide alarm data, historical data, and real-time analog/status values. Original OPC is now called OPC Classic, and the new version is OPC-UA.

OPC-UA: A re-engineered version of OPC that is platform-independent and does not rely on the use of Microsoft Windows and its DCOM support. OPC-UA was developed as a replacement for the original OPC after Microsoft announced the discontinuation of DCOM and RPC, both of which were required by OPC classic.

Open-loop control: A control sequence, from input measurement to control action, that requires human interaction (as opposed to closed-loop control, in which control actions are automatic and autonomous of human intervention).

Operator console: A computer workstation or PC that provides operational displays. Alarms, trends, and other information required by pipeline operational personnel in order to perform their jobs, including the ability to initiate supervisory control actions.

Out-of-band signaling: Messages, commands, or data transmitted with the primary data stream that are considered a control signal and demand immediate attention. The receiving side must pass on the out-of-band data to the appropriate software routine in front of any other data that has been buffered and not yet processed, because the command must be executed as soon as possible.

Overview display: The highest-level display (least detail/greatest coverage) in a display hierarchy. Generally, a geographic map–style display with poke points for navigation down to more detailed display levels, as well as indicators/icons with summary information.

Pairing: The (semi-)automated process of two Bluetooth-enabled devices forming a persistent connection for data/voice exchange purposes.

Packet: A message sent across a packet-switching network with the necessary header/trailer information, per the protocol being used, and carrying a payload of data. (Similar to a *datagram*.)

Packer filter: A type of basic firewall that passes or blocks messages attempting to transit in either direction based on a set of firewall rules and the data contained in the various message headers.

Packet sniffer: A software package or diagnostic appliance that monitors message traffic over a network (particularly a wired LAN or wireless LAN) and allows the capture and display of messages meeting defined criteria. A packet sniffer can be a useful diagnostic tool but is a dangerous hacker tool as well.

Packet switching: A type of WAN that breaks messages into small, limited-sized pieces and then sends them each as a separate data packet, possibly via different routing, through the network to the destination, where they are collected, put into proper order, and delivered as a complete message. X.25 frame relay and IP networks are packet-switching networks.

PAN: Personal Area network.

Pass-through mode: Allowing messages on one port of a device such as an RTU to be repeated on a different port, creating the effect of eliminating the device from the resultant circuit. Like a port switch. (Also called *transparent mode.*)

Password cracker: A hacker tool/utility program that attempts to break (guess) a system/user password by sending all possible character variations, one at a time, or by sending character strings selected from any number of word lists (see *dictionary attack*). These programs are effective only against systems that place no limit on the allowable number of login attempts made by a user.

Penetration (pen) testing: Attempting to break into a specified system, using hacker tools and techniques, with the permission of the system owner, to identify security vulnerabilities. Pen testing may be a red team (black box) test or a blue team (gray box) test, based on whether some or no insider information is made available about the system to be tested.

Peripheral transfer switch: An electronic device used to share/switch a peripheral device between a redundant set of computers. Usually part of an automatic "fail-over scheme," transferring peripherals from a previously functioning (now failed) computer to the backup computer. Needed when peripherals cannot be shared on a common bus (as an EIA-232 circuit) or do not support dual connections (as a printer with a parallel port).

Piconet: The term used to describe a wireless network of Bluetooth-connected devices that are all with a small physical distance from each other.

PID algorithm: Proportional-integral-derivative calculation. A third-order differential equation used to perform continuous regulatory control in computer-based process automation systems. Emulates the actions of analog circuits used for regulatory control prior to the introduction of digital computers.

Ping: An Internet utility program that verifies whether a specific IP address (or URL) is accessible, using ICMP messages. Also provides an indication of the time required for a round-trip to/from the specified IP address. Ping will make a URL lookup call to the DNS if given a URL, rather than an IP address.

Ping sweeping: Going through a range of destination IP addresses, one at a time, and sending a ping to each such address in an attempt to identify the presence of active computers with those IP address assignments. (Similar to network scanning, except only the ICMP messages and not others such as TCP and UDP.)

PKI: Public-key infrastructure. Related to public-key encryption, in which a trusted third party (the CA) issues electronic credentials (digital certificates) that include public keys used for encryption and subsequently validated by the CA.

Plaintext: Original form of a message, prior to conversion into ciphertext.

PLC: (1) Programmable logic controller. A microprocessor-based device with I/O hardware and user-defined control logic that cyclically reads the inputs, computes the resulting logic, and then drives the outputs to the desired state. A PLC will generally support some form of communications and various protocols. Widely used in the water/wastewater industry as RTUs. (2) Power line carrier. A low-bandwidth communications technology used by electric utilities, in which messages are sent over the transmission line conductors that carry electric power.

Point: A single physical I/O or calculated value in the real-time database of a SCADA system or DCS. (Also called a *tag*.)

Point tagging: Placement of one or more software flags on a control point in a SCADA system or DCS to inhibit the sending of control commands to (or the discarding of received commands by) that specific output (analog, pulse, or contact). Analogous to the physical placement of paper tags on the control handles of switch gear in a control room. This safety feature prevents the SCADA system or operator from (de)activating or adjusting a control output while the associated process and/or equipment is in an unsafe condition.

Poke point: A place/area/icon/object on a display screen that, if selected with the pointing device (mouse, touch screen, etc.), will cause a predefined software action to occur.

Polling: Interrogation of RTUs for data updates through the various communication lines and protocol addresses supported, to cycle through all RTUs on a regular basis.

Port scanner: Software utility/hacker tool that sends specially crafted IP messages to a designated IP address, to identify what TCP and/or UDP ports are recognized and supported by the computer system at that IP address. Used to identify known vulnerabilities in various operating system versions.

Port switch: A device that allows a serial communication circuit to be connected to several devices one at a time, through user commands that select the desired device.

Private key: In PKI, the user has a pair of mathematically related keys (a private key and a public key), both of which can be used for encryption and decryption. The public key can (and should) be made available to anyone who needs

communications security. The private key must never be revealed, or the security of the encryption scheme will be compromised.

Privatization: Placing the support and maintenance of a formerly governmental/public activity in the hands of a private company that will assume responsibility in exchange for financial compensation.

Protocol: A set of fully specified message formats, data formats, and rules that define the mechanisms for establishing, managing, verifying, and terminating data exchanges between two computers/devices.

Protocol analyzer: A software package that allows a separate computer to be used to monitor, record, and display the messages sent over a serial communication channel or LAN. (See also *packet sniffer* and *protocol test set.*)

Protocol-aware: In reference to a protocol analyzer, this means that the analyzer can dissect the protocol messages into their individual fields and data elements. In reference to a firewall, this means that the firewall rules can include checks that reference protocol message individual field values and data elements.

Protocol converter: A stand-alone computer, software package, or communications appliance programmed to use one protocol on one port and different protocol(s) on the other port(s) and to unpack and then repack the data and commands passed between the ports. RTU protocol converters allow a different protocol to be used without having to change the protocol running in the existing RTU or host equipment.

Protocol stack: The layers of software that together implement all of the logic necessary to communicate with a similarly equipped computer or communications device running the same protocol. Depending on the protocol, the stack might be only two layers (as in serial Modbus) or up to seven layers (as in the ISO/OSI model).

Protocol test set: Similar to a protocol analyzer except programmed with more specific knowledge of the protocol(s) monitored, with information and displays tailored to the unique features and nomenclature of the particular protocol(s).

Proxy server: A computer used to isolate applications from the actual server they are attempting to access, for security purposes and for performance improvements. If used as a firewall, the proxy server restricts message traffic to a specific set of protocols and processes the protocol messages and then allows their passage if they meet security requirements. The proxy server is protocol-aware, meaning that it can check a message for conformance to protocol specifications.

Pseudo I/O point: A point in the real-time database of a SCADA system or DCS whose value is set by manual input or application programs, but is processed, alarmed, and treated in the same manner as physical I/O signals.

PSTN: Public switched telephone network. The wires-based dial-up telephone system used for noncellular telephone calls, established by the Bell Telephone Company. A telephone line used to connect to the PSTN may informally be called a POTS (plain old telephone system) line.

Public key: The encryption key, of a PKI key pair, that is disclosed to others so that they may encrypt messages such that only the holder of the corresponding private key can decrypt them. (See also *private key.*)

Publish-subscribe data delivery: An unsolicited report-on-change data-delivery mechanism, allowing producers of information and consumers of that information to be linked for the delivery of new information as it becomes available.

PVC: Permanent virtual circuit. A logical point-to-point communication connection, established across a frame-relay network, that persists between messages and does not have to be reestablished at the start of each transaction.

Raw count: The binary value generated by an A2D circuit for a given value of input voltage. All A2D circuits produce binary values, which need to be translated into the engineering units of the physical parameters measured (pressure, temperature, level, etc.).

Real-time: Actions occurring as a result of some event and which are occurring with minimal time delay from the time of the initiating event.

Real-time clock: The circuit in a computer that generates a hardware interrupt at a well-defined and consistent time interval. These intervals can be counted and used to compute elapsed time, as well as time of day. The real-time clock in a computer needs to tick at a rate that is substantially faster than the fastest event that must be dealt with by the computer programming (e.g., moving a cursor, detecting keyboard strikes, or reading in the next character received on a serial port).

Reed relay: An electrical component composed of two thin iron strips, in near proximity, enclosed in a glass bulb that is wrapped with a coil of wire such that by putting a voltage on the wire coil, the resulting magnetic field causes the two metal strips to become magnetically attractive and thus bend together to form a circuit.

Remote exploit: A series of IP messages crafted to interact with applications that accept connections (e.g., services) in order to reach a point where a malicious message can be sent to perform a buffer overflow on the service.

Repeater: A device that receives a weak and possibly distorted signal and regenerates that signal in an amplified and undistorted manner to allow the signal to propagate further.

Report by exception: A communications protocol used in RTU polling in which message traffic/time is reduced by allowing the RTU to report only those details that have changed since the prior polling, rather than sending a full report of all data values (many of which may not have changed) with each poll. Since additional identification information is needed, there is a crossover point at which a full report becomes more efficient than report by exception.

RFID: Radio Frequency ID. A bar code-like electronic tag that can be attached to items and read from a distance (ranging from inches to almost 100 feet). The tag can contain a simple, fixed numeric code, or it may be rewritable and carry

a moderate amount of information. Tags are normally read by activating them with the RF signal of the reader. Noncontact ID badge readers and electronic toll booth scanners are examples of RFID technology. RFID tags are often used for inventory tracking and routing in automated warehouses.

RLL: Relay ladder logic. A graphical representation of Boolean logic, originally designed to describe actual electromechanical relay circuit wiring used for control logic and later adopted as a programming language for use with PLCs (which replaced electromechanical relays).

Rogue AP: An unauthorized Wi-Fi access point attached to a LAN, usually without enabling adequate protective measures.

Rollback: Restoring the prior version of software (or files/data) when installation of an update goes badly or results in performance or reliability reductions. When a SCADA system (or major application) is upgraded to a new version, safety procedures normally include preparing to revert to the previously upgraded version of the software.

Router: A communications appliance that interfaces between a local network and one or more communication links to other networks, for the purpose of sending packets destined for computers on these other networks via the proper link (and receiving packets and placing them onto the local network). Routers usually provide a buffering capability and may perform added functions (e.g., packet filter/firewall).

RTD: Resistive temperature device, an electrical component that has a resistive value that varies by the temperature to which the device is exposed.

RTO: Regional Transmission Organization. Under deregulation of electric utilities, these organizations (along with ISOs) were established to ensure a competitive market for power generation and transmission.

RTU: Remote terminal unit. An electronic device (now microprocessor-based) with I/O hardware and a communications interface, placed at a remote process location, so that a SCADA system can communicate with the device, to receive updating information about the process and to send commands that effect the process.

Rule base: The set of rules in a firewall that determines what packets will be permitted to transit the firewall and what packets will be blocked. A.k.a. access control list (ACL).

SAM file: The Security Account Manager (SAM) is a database file in Windows XP, Windows Vista, Windows 7, 8.1, and 10 that stores users' IDs and password hashes. It can be used to authenticate local and remote users, but beginning with Windows 2000 SP4, Active Directory contains those hashes and authenticates remote users. Hashes can be stored in either/both LM hashes or NTLM hashes. (For Linux, see Shadow file.)

Sandbox: A controlled, possibly virtual, environment within which software can be allowed to execute but which strictly limits any programmatic actions that

could be dangerous or malicious, e.g., malware may be given a heuristic analysis by running it in a sandbox environment.

SCADA: Supervisory control and data acquisition. A type of computer automation system that uses a central computer system, wide-area communications technologies, and a large number of geographically distributed RTUs to monitor and control geographically distributed processes such as pipelines and electric power transmission.

Script kiddie: A person who lacks the advanced computer expertise and skills of a hacker but employs easy-to-use, readily available hacker tools to launch attacks. A derogatory term used by hackers/crackers, although some use the script kiddies to do their dirty work.

Scripting language: A programming language that is interpretive in nature and that is executed by a suitable application program (an interpreter), rather than being converted into actual computer instructions and executed directly by the computer. VBScript, PHP, and JavaScript are prime examples.

Sequence of events: The recording of multiple associated events with associated time tags, usually from multiple sources, so that it is possible to determine which event was the initiating event and which events followed, in what order and timing.

Session key: An encryption key generated specifically for one-time use, to protect message traffic during a communications session. A session key strategy prevents an attacker from collecting enough message traffic based on one key to break the encryption. (See also *key.*)

Set point: The numeric value that represents the desired value of a controlled parameter (e.g., flow, level, or pressure) in an automatic control system. The set point defines what the controlled parameter should be, and if it drifts away from the set point, the control system will make control changes to return the parameter to the value of the set point. A PID loop is used for this control purpose.

SFC: Sequential Function Chart. Graphical programming language used to design sequence logic, similar to RLL. SFC is one of the five languages defined by the IEC61131-3 standard. The SFC standard itself is defined in IEC848 ("Preparation of function charts for control systems") and was based on GRAFCET, a graphical programming tool developed in Europe. Most PLCs and many modern RTUs support SFC programming tools.

Shadow file: In a Linux system, the Shadow file contains user IDs and hashes of user passwords, used for authentication purposes. The file supports advanced encryption algorithms and is accessible only to a user with root access.

Shell: Term used for a command-line interpreter window or process on a Linux system that allows a user to enter Linux commands to a CLI such as BASH or KSH or ZSH.

S-HTTP: Secure Hypertext Transfer Protocol. A secure alternative to the HTTPS protocol that incorporates client-server authentication and message encryption functions. (See also *HTTP* and *HTTP/S.*)

Simplex channel: A communication channel that supports data transmission only in one direction. Broadcast radio and television are examples of simplex communications.

Smart radio: Radio equipment with integral microprocessor technology such that the radio can be treated as a computer and can communicate via serial or Ethernet interfaces and protocols. The radios in a communicating group can send each other out-of-band messages, for control of the flow of traffic and for error checking, coordination, and message transmission retry purposes.

Social engineering: Exploiting typical human behavior to trick people into giving out confidential information, violating security policies and procedures, or providing improper access to secure areas or systems. Social engineering is used by threat agents to gain the insider information they need to launch an attack.

Software library: A set of pretested and documented software modules or utilities that can be used by a programmer as part of their application development. In a SCADA system, such a library might include utilities for sending commands to RTUs, as well as subroutines for presenting information on HMI screens and for manipulating alarm management settings.

Solar panel: A photovoltaic panel that generates electrical power when exposed to sunshine. Solar panels are used to power RTUs in remote locations where no other power source is available. Such RTUs usually include a battery charger that supplies power when sunshine isn't available, and power-down circuitry to minimize power usage between polls from the SCADA system.

Solar-powered: Equipment (e.g., RTUs and radios) designed to operate at the low power levels afforded by solar panels, which use the energy in light to generate electricity.

SONET: Synchronous optical networking. A high-bandwidth telecommunication standard based on fiber-optic transmission, capable of carrying vast numbers of simultaneous telephone conversations or high-speed data. SONET was designed to be highly deterministic and is thus suitable for such data-streaming applications as video conferencing. Supported in the United States and Canada but supplanted by SDH in the rest of the world. Can be used as the basis for ATM networking.

SOX: Sarbanes-Oxley Act of 2002. Mandates the accuracy and reliability of corporate financial disclosures and imposes information security and protection measures directly related to corporate cybersecurity issues of information confidentiality, integrity, and availability. Extends to any business process-related data, which can include some of the types of data maintained in SCADA systems.

SPA: Stateful protocol analysis. A firewall capability in more advanced firewalls in which the firewall checks that the message sequences for any specific protocol

(typically application layer protocols) follow the "rules" for that protocol. It is the process of comparing predetermined profiles of generally accepted definitions of benign protocol activity for each protocol state against observed events to identify deviations.

Spam: Unsolicited (and undesired) emails, particularly of a marketing or advertisement nature, that are normally sent out in bulk (to a huge number of recipients), often on a regular basis. The email equivalent of junk mail.

Spread spectrum: A method of making a radio signal less susceptible to jamming or interference by employing either signal spreading, using spread spectrum mechanisms that make the signal appear over a wide frequency range, or channel hopping, where the participants switch from channel to channel on a pre-defined pattern while transmitting.

SQL: Standard Query Language. An ANSI/ISO standard language for manipulating relational databases that is supported by all major relational database vendors.

SQL injection: Attacking poorly designed and programmed Web sites with back-end relational databases by entering specially crafted strings into the data entry fields of Web page forms, so that these strings are relayed to the back-end relational database and accepted/executed as valid SQL commands.

SSID: (Wi-Fi) Service Set Identifier. A code (32 characters or less) used by a Wi-Fi access point (AP) and all of the communicating devices to identify them as part of the service set sharing the corresponding AP. Normally, an AP broadcasts its SSID, but this can be disabled as part of an overall security strategy (a strategy that should also include encryption and authentication and possibly MAC address blocking).

Stack overflow: See *buffer overflow.*

Stateful inspection firewall: A firewall that knows the allowed states and state transitions for TCP sessions (as well as the reasonable times allowed for transition between certain states) and the state transitions permitted from each state. A stateful inspection firewall will reject/cancel connections that violate proper message/transition/timing rules, preventing certain types of attacks (e.g., SYN flooding) from being successful. (See also *deep inspection firewall* and *stateful protocol inspection.*)

Static IP Address: The assignment to a computer or device via configuration or setup files so the IP address (and associated information) is present at any reboot or restart of the computer with no dependence on an external mechanism to provide this information.

Strip-chart recorder: An electromechanical recording device that uses a moving strip of paper to record the movement of a pen whose position is controlled by a voltage source that represents a process measurement.

Strong authentication: The use of multiple factors to determine/validate the identity of a user, such as a password and either a magnetic-strip ID card (or RFID) or a biometric measurement.

Strong encryption: Performing encryption with a key that has a sufficient size such that a brute force attempt to determine the key, even with a huge number of cooperating computing platforms participating, would still take so long that any value in recovering the encrypted information would have been lost. A.k.a. cryptographically strong.

Structured walkthrough: A static testing technique performed in an organized manner between a group of stakeholders to review and discuss the aspects of a formal procedure or process. The main objective in a structured walkthrough is to find defects, errors, or functional problems in the procedure so that they can be eliminated.

Substation automation: The use of a computer-based device in an electrical substation to both collect data from most/all of the IEDs in the substation so that this data can be provided to a SCADA system via a single connection; and to enable remote access to those IEDs for maintenance and diagnostic purposes. A.k.a. substation data concentrator.

T1/T3: Telephone system classifications of communication circuits designed to carry a given number of voice telephone channels by being time-division multiplexed. A T1 (or DS1) circuit runs at 1.544 Mbps (24 voice channels). A T3 (or DS3) circuit runs at 43 Mbps (672 voice channels).

Tag: See *point.*

Tag name: The reasonably short descriptive text string assigned to a point (tag) in a SCADA system or DCS, to aid the operator in recognizing that particular point and its physical nature and location. Often based on a standardized company naming scheme or on the naming scheme proposed by the ISA.

TCP: Transmission Control Protocol. A protocol that runs over IP, providing reliable message delivery and stream-like connectivity and allowing the use of ports.

TCP fingerprinting: Identifying an operating system running on a remote computer by analyzing the packets it sends and how it responds to specially crafted packets. Nmap (Network Mapper) is a commonly used Unix tool that performs this function.

TCP/IP: A commonly used conjunction of Transmission Control Protocol and Internet Protocol. Part of the overall IP suite of protocols.

TDM: Time-division multiplexing. Subdividing (sharing) a single communication channel by allocating it on the basis of time slots whereby any application can use the entire bandwidth for a limited amount of time. The telephone companies subdivide their communications systems by using time-division multiplexing.

Terminal server: A computer-based device that connects to a TCP/IP network and provides a local serial, asynchronous (RS-232) communication circuit that is simulating a metallic circuit to a remote computer's virtualized COM: port so that the remote device and computer can communicate using serial protocols or basic ASCII character messaging.

Thermocouple: An electrical component composed of two metals bound together and which, when exposed to different temperatures, generates a corresponding varying millivoltage signal.

Thermoelectric generation: Generating electrical power in a device by exploiting the thermoelectric effect, which dictates that physically joined dissimilar metals, when heated, generate a small electrical potential (the principle used in thermocouples). Thermoelectric generation is very reliable, owing to the lack of any moving parts, and is often used as a power source when fuel is available but electricity isn't.

Totalizer: A counter that counts up due to some form of input signal and where each increment of the counter represents some physical quantity (e.g., gallon, kWh, cubic foot, etc.) such that the totalizer can be used to determine both total quantity and changes of quantity between each reading. A.k.a. *accumulator*.

Transmitter: In process instrumentation terms, a device that generates a variable signal such as variable air pressure or variable electrical current, corresponding to the value of a process measurement such as temperature or pressure. Also referring to an electronic device that generates a radio signal that is modulated in some manner so as to convey information.

Transport mode: See *pass-through mode.*

Trojan: An apparently useful/benign software application that contains a set of logic that is triggered under the proper conditions and that may cause damage to a computer system, open up a backdoor for an attacker, send out confidential information to a remote computer, or perform some other undesirable (and unauthorized) activity. Also called *Trojan horse,* alluding to the tactic used by the Greeks during the Peloponnesian (Trojan) War.

TTL: Time to live. In every IP packet, a counter that was initially intended as an actual time but is now used as a hop counter. Every time a packet goes from one computer to another (in an IP network), the count is decrementally adjusted. If it reaches zero, the packet is discarded and an ICMP message is sent to the sender of the message. Network mapping can be done by sending out IP packets with TTLs that start at one and are adjusted incrementally, tracking each computer that subsequently sends back an ICMP message.

UART: Universal asynchronous receiver transmitter. An electronic circuit that performs all of the hardware tasks required for asynchronous serial transmission or reception of a data octet/character, including adding and removing start and stop bits (used to signal the start and end of an octet frame) and bit sampling and storage at the user-specified transmission rate.

UCA2.0: Utility Communications Architecture. An object-based, high-performance architecture originally used within electrical substations for inter-IED communications (on an Ethernet LAN), also used across IP networks and adopted by some municipal water/wastewater utilities.

UDP: User Datagram Protocol. One of the two transport protocols used to connect applications via IP networking (see also *TCP*). Adds the concept of a port to IP networking. A less robust communication transport mechanism than TCP.

Unbalanced: See *asymmetric*.

UHF: Ultrahigh frequency. Radio frequency band allocated above VHF and below superhigh frequency, with a range of 0.3 to 300 gigahertz. Used for data exchange and wireless communications, as well as cell phones and television.

Unix process: In Unix/Linux, a program or task that is running under the control of the operating system. Assigned a process ID (pid) and allocated resources as required and authorized on the basis of priority.

URL: Universal resource locator. The text string entered into a Web browser or email to direct it to the desired computer, application, and network. A URL is more user-friendly than remembering an IP address, although a URL is actually converted into an IP address (and port number) by the DNS. A URL has a protocol designation, a resource and domain portion, and any added information needed to specifically identify the target.

Validation: Verifying that a user or an application program has the authority to perform a requested operation.

VHF: Very high frequency. Radio frequency band allocated from 30 to 300 megahertz. Used by some radio communications equipment found in older SCADA systems because of superior propagation properties.

Virtual circuit: Simulation of a dedicated, circuit-switched, communication path through a packet-switching WAN. A frame-relay network allows for the creation of persistent (permanent) virtual circuits that connect end point equipment.

Virtual RTU: The software running in some computer-based device or system that emulates the slave portion of an RTU polling protocol and that is used to provide the data in that device or system to a SCADA system as if it were an actual RTU.

Virus: Malware that spreads by making copies of itself and appending them to files, documents, and the boot sectors of removable media. A virus may demonstrate multiple phases, once activated, the first being self-replication and the next being infliction of damage.

Virus signature: Portions of computer machine code that are part of the overall virus program. A firewall or virus scanner maintains a table of such signatures, which are sufficient to identify particular viruses, and searches for their presence within IP message packets or appended to other programs.

VLAN: Virtual LAN. Subnetwork of interconnected nodes within a larger physical LAN/WAN, with switches and routers configured to restrict routing among only specified sets of end points.

Virtual Machine (VM): A virtual machine is a computer file, typically called an image, that when executed, behaves like an actual computer. In other words,

creating what seems like a computer within a computer. It runs in a window, much like any other program, giving the end user the same experience on the virtual machine as they would have on the host operating system itself. VMs allow you to preserve legacy environments and applications and simulate legacy computer systems.

VNC: Virtual network console; a form of remote desktop that permits the desktop of a machine to appear on your local machine as if you were sitting in front of the other computer. Very useful for accessing "headless" computers/servers. Client (viewer) and Server software available for both Windows and Linux.

Voice-grade circuit: A communication circuit (e.g., telephone line or radio link) with sufficient continuous bandwidth (typically 64 kbps) to deliver the digitized midrange frequencies of the human voice.

VoIP: Voice over IP. Technology that provides real-time voice-grade telephone communications across an underlying IP network, using protocols designed for such traffic.

VPN: Virtual private network. A set of interacting computers on a larger network that ignore all other computers on that larger network and accept and respond to communications only from those computers that are part of the VPN. Also used to describe the use of authentication and encryption to secure the communications between a central computer and a remote (possibly mobile) PC (see *VPN tunnel*).

VPN tunnel: Establishing a point-to-point connection between two computer devices, or suitably equipped communication devices, such that the two devices can authenticate, exchange encryption keys, and then transport all message traffic between these two devices as encrypted packets. This may include taking non-secure IPv4 packets and transporting them as data in encrypted IPv6 packets. The two most common uses are to provide a secure interconnection of two geographically separate LANs and to allow a mobile or remote user secure connectivity to the corporate network, over a dial-up or Internet connection.

Wallpaper: A static image placed on the display screen in the background, on which dynamic display elements are placed and updated. In a SCADA HMI, this could be a map image or the top-down architectural layout of a facility.

WAN: Wide-area network. A network that spans a geographical area too large to be served by PAN, LAN, or MAN technologies. Where a WAN starts and a MAN ends is not well defined.

War dialing: Using a daemon dialer program to automatically call consecutive telephone numbers within a user-specified range, recording those telephone numbers at which active modems responded. A.k.a. daemon dialers.

War driving: Traveling through a geographic area (often an urban or suburban area) with a laptop PC equipped with a Wi-Fi receiver and software that identifies wireless access points (APs) and their characteristics, recording these locations, often to add to a database on a hacker Web site.

Watchdog timer: An electronic circuit designed to send a reset/reboot signal to a computer if the circuit is not sent appropriate commands within a predefined time window. Used to automatically reboot computers that have crashed or are hung (in an infinite loop).

Webcam: Web camera. A self-contained video camera that operates as an autonomous Web site/server and that supplies updating video images as HTML pages to any conventional Web browser program.

Web page: A hypertext document (usually HTML, XHTML, or XML) offered for delivery and display (across a network using HTTP) by a Web server, to any suitably equipped Web client (a Web browser program).

Well-known port numbers: TCP and UDP port numbers in the range 1 to 1024 that have been officially assigned to common services used in most servers such as a Web server, DHCP services, DNS services, email servers, etc.

WEP: Wired Equivalent Privacy. A 40 bit encryption scheme that has fallen out of favor due to the ease with which it can be broken by commonly available hacker tools such as aircrack-nt.

Whitelisting: A mechanism for setting a list of approved items (Web sites, application programs) and then blocking any other items not on the list.

Wi-Fi: Wireless Ethernet technology.

Wi-Fi hot spot: See *hot spot.*

WLAN: Wireless LAN. A LAN based on some version of wireless Ethernet (Wi-Fi/WiMAX) technology. Not typically used in reference to other wireless networking technology (e.g., Bluetooth or ZigBee) although that distinction is blurring.

Worm: Malware specifically designed to spread across network connections, rather than by transport through other programs like a virus or Trojan.

X11: The protocol defined at the Massachusetts Institute of Technology for messages sent between an X-Window client/server (X-terminal), describing the display to be rendered and returning keyboard and mouse events for processing.

X.25 packet switching: The earliest type of value-added, commercial, WAN packet-switching technology, to network large numbers of dumb terminals to central computers. X.25 packet switching still exists but is being supplanted by frame relay, partly because of the replacement of dumb terminals by computer-based devices.

X.509: An ISO standard that defines the format and information content of a digital certificate for use in a PKI authentication scheme. (The current version is X.509v3.)

XML: The extensible markup language. A very popular markup language in B2B data exchanges, in which users can define their own tags and create self-describing data in the form of a Web page.

X-terminal: A computer display terminal device (or a PC running software emulation) that implements the X-Window standard and supports the X11 protocol. In Unix, this is a dumb terminal but with a graphical, windowed interface, allowing a user to interact with a central server.

Zenmap: A version of the popular NMAP network scanning utility that incorporates a windowed GUI and several useful data display modes. (A user-friendly version of NMAP.)

ZigBee: Reliable, low-cost, low-power wireless networking technology intended for industrial monitoring and control as an M2M technology.

Zombie computer: A computer (normally with a full-time Internet connection) that has been infected with a Trojan or timebomb such that it can be used (along with thousands of others) to initiate DDoS attacks on designated targets, either at a specific time/date (timebomb) or upon receiving the necessary command (Trojan). It is estimated that there are at least five distinct sets of zombie computers around the Internet, controlled by various hacker groups, that have been used to launch such DDoS attacks.

Index

A

A2D converter, 25, 35, 46, 70, 450, 452
access authority, 289
access control list (ACL), 115, 231, 283, 285, 318, 443, 466
access controls
    electronic, 283, 289
    hybrid, 282
    manual, 280, 282
    to operating systems, 179, 208, 210
    to SCADA system utilities, 104-105, 115, 281
    to secure areas, procedures for, 280-282, 303
accessorization, 91
access rights, 102-104, 106-107, 208, 213, 230-231, 268, 279, 282, 290, 301, 353, 433-434
access tracking, 287
access verification checks, 118
account, 102-103, 105, 107, 130-132, 204, 208-210, 212, 219, 231, 233-234, 242, 249, 256, 268, 270, 273, 276, 304, 311, 340, 350-351, 403, 410, 412, 417, 427-428, 430, 433-434, 466
account lockout, 351
account management procedures, 103-104, 301
accumulator freeze, 43
accumulators, 28, 44
acknowledged alarms, 150
Active X Controls, 274-275, 461
addressing scheme, 74, 85, 179
address mapping, 81
address resolution protocol (ARP), 90, 196, 200, 444
ad hoc operating modes, 192, 194, 445
administrative programs, 208, 319
administrative rights, 103, 208
agents, 229-231, 251, 289, 309, 311, 320, 330, 333, 341, 395-396, 422, 459, 468
AirSnort, 329
alarm acknowledgments, 153, 443
alarm annunciation, 108, 152
alarm filtering, 151, 443
alarm history file, 152
alarming functions, 114, 148-149, 211, 305
alarm limit checking, 149
alarm-state visual indication, 148, 153
allowable-value range checks, 149
alternative (backup) facilities
    application programming done on, 103, 170
    for availability, 170
    in centralized architecture design, 11
    insider knowledge of, 171, 393
    point-to-point connections with SCADA, 73, 252
    restoration from, 251-252, 290, 412, 418
alternative operating site, 58, 313, 364
American Gas Association (AGA), 425
    volumetric computations, 30
analog circuits, 64-65, 462
analog circuits vs. digital circuits, 64, 371
analog input point, 149
analog inputs, 24-25, 28, 35, 40-41, 109, 149-150
analog modems, 64, 73, 75, 378
analog outputs, 26, 40
analog telephone circuits, 23, 379
analog-to-digital (A2D) converters, 25, 443
annual review, 299, 308-309, 411-412
anomalous traffic, 175-176, 314, 326, 328, 410, 455, 460

anomaly-detection technology, 191
application development, 298, 468
application-layer security, 92, 101, 126, 313, 315, 469
application program interfaces (APIs), 49, 444
application programming library functions, 118
application proxy, 315, 449, 453
application proxy server, 453
APs, 174, 238, 266, 436-437, 440, 473
ARPANET, 178
ASCII text file, 448
association control service element (ASCE), 372
association of control elements (ASCE), 126
asymmetric encryption, 218
asynchronous protocols, 4, 444
asynchronous transfer mode (ATM), 83-84, 444
attackers, xxi, 54, 72, 93, 99, 102, 132, 140, 152, 155, 167, 171-172, 174-175, 177, 188, 191-193, 200, 202, 210, 215-216, 219, 225, 229, 231-234, 236, 238, 242, 249, 251-257, 265, 268-270, 272-277, 279, 283, 288-290, 292-293, 295-296, 306-307, 321, 323, 325-326, 344-347, 350, 354-357, 361, 366, 368, 371-372, 384-385, 388, 395, 398-399, 404, 406-407, 411, 436-438, 446, 452, 460, 467, 471
attack message classes, 188
attacks. *See also* cyber attacks and cyber threats
  impact of successful, xxi, 54, 132, 306, 398-400
  most probable objectives from, 99, 188
  probability of, 17, 343, 384, 397
  and vulnerability points, 163, 238, 244
audible signals, 152
audit files retention procedures, 303
audit trail, 104, 131, 153, 164, 282, 289, 299, 303, 412
authentication, 223, 285, 331, 348, 350, 353, 361, 368-369, 373-374, 384, 394, 417, 444-446, 474
authentication technologies, 211, 220, 222, 283, 371-372, 438
authentication/validation process, 131, 208, 210-211, 213, 218, 221, 276, 282, 303, 347, 424, 445, 457-458, 467-469, 473
auto-answer modems, 67, 173
auto boot function, 169, 190, 205, 212, 305, 360, 397, 451, 469, 474
automatically generated standard displays, 136
automatic mode control loops, 213, 244, 284, 390
automatic schemes, 9, 284
autorun function, 350
availability, 17, 43, 57, 80, 101, 123, 169-171, 174, 330, 365, 423, 468

B

backbone, 68, 83, 94-96, 185, 243, 392-393, 395
back doors, 281, 284, 288
backdoors, 102, 116, 131, 191, 231, 261, 267, 276, 278, 366, 368-369, 373, 400-401, 410, 445, 471
background checks, 239, 301, 309, 428, 434
backup computers. *See* alternative (backup) facilities
backup SCADA systems. *See* alternative (backup) facilities
bacteria, 276, 445
bandwidths, 5, 17, 39, 54-56, 58, 62, 65, 67, 69-70, 72, 74, 78, 81, 83-84, 95-96, 127, 198, 203, 296, 326-327, 370, 373, 379, 381, 386, 391, 395, 401, 423-424, 444, 451-453, 457, 463, 468, 470, 473
baselines, 169, 412, 428, 445
basic protocols, 89
battery backup, 43, 445
baud rates, 109, 424
Bell Labs, 69
bilateral agreement, 373
biometrics and biometric devices, 131, 210, 282-284, 438, 452-453, 460, 469
bit-oriented message format, 4-5
bit-oriented protocol, 4-5, 445
blackouts, 188, 305, 366, 399
block ciphers, 215-216
Bluesnarfing, 260
Bluetooth ad hoc networks, 462, 474
Bluetooth-enabled devices
  electronics system security and, 76, 194, 428, 435-436, 445, 461
  spread-spectrum radio equipment, 76

and viruses, 35, 194, 435
as vulnerability points, 194
and worms, 194
Bluetooth wireless capability, 34, 76, 194, 436, 445, 462, 474
Boolean logic, 41, 445, 466
bot networks, 191
breaking the encryption, 215, 266, 467
bridging, 193, 360, 444
Bristol Standard Asynchronous/Synchronous Protocol (BSAP), 54
broadcast messages, 44, 90, 198-200, 444
broadcast radio, 67, 392, 468
brute force approach, 210, 216, 267, 347, 470
buffer overflow, 329, 331, 445, 465
buffer overflow attack, 175, 268-270
buffers, 191, 197
bump-in-the wire encryption devices, 51, 384
bump-less schemes, 12
business-to-business (B2B) Web servers, 271, 444, 474

C

calculated point utility, 110-111
calculated values, 20, 109-110, 148, 152, 156, 165-166, 463
call forwarding, 62
call-forwarding techniques, 62, 348
cathode-ray tube (CRT), 6, 129, 213
CDMA enabled cellular modems, 80
cell phones, 35, 61, 62, 64, 70, 71, 76, 78, 79, 80, 152, 172, 194, 206, 210, 216, 233, 279, 298, 427, 445, 472. *See also* mobile electronics
cell phone text messaging, 152
cellular communications, 80
cellular modems, 79-80, 378
central access library, 115
central computer, 1-3, 10, 23-24, 372, 384, 467, 473-474
centralized operator-console architecture, 10
centralized user authentication, 203-204, 210, 284, 353
centralized Web-based HMI architecture, 35
central processing unit (CPU) resources, 9, 12, 23, 29, 48, 103-104, 116, 119, 270-271, 316, 424, 445, 447, 450, 459
certificate authority (CA), 222-223, 445
challenge handshake authentication protocol (CHAP), 131, 446
challenge-response scheme, 131, 210, 446
channel service unit/digital service unit (CSU/DSU), 448
character-oriented protocols, 4, 446
check-before-operate sequence, 9, 446
check code, 53-54, 170, 221
checksum, 53
chipping code, 76, 451
cipher, 214-217, 220, 282, 446, 458
ciphertext, 171, 214-215, 220, 446, 449, 458, 463
circular buffer, 158, 446
circular-chart recorders, 157
Cisco, 184, 200-202, 241, 257, 259, 261, 263, 265, 267, 320, 329, 346, 403, 407-408
clients, 14, 188, 207, 268, 312, 315, 327, 334, 336, 342, 347
client-server designs, 14, 17, 126
clocks, 44-45, 341
code letters for measurement conditions, 154
cold-backup schemes, 12
collection systems, 50, 328, 367
collision recovery, 53
color coding, 113, 154, 228
command line interpreters (CLIs), 31, 103, 178, 191, 238, 320, 446-447, 467
comma-separated value (CSV), 111, 448
comma-separated value (CSV) file, 111, 448
commercial hardware and software vulnerabilities
DCOM, 122-123, 126, 255, 349
firewalls, 89, 175-176, 180, 182-183, 249, 277, 325
OPC, 122, 126
operating systems, 101, 116, 241, 349, 403
relational databases, 112, 406
TCP/IP, 269
web site/server, 191, 403-405, 407
X-Windows, 405
commercial off-the-shelf (COTS) hardware, 86, 116, 179, 349
commercial software
as hacking tools, 189, 357
IP networking with, 55
for programmable logic controllers (PLCs), xxii, 463

SCADA software evolution with, 101
SCADA systems as, 16, 406-408
vulnerabilities of, 123, 126, 189, 241
commercial voice/data carriers, 69-70
Common Object Request Broker Architecture (CORBA), 125, 447
communication modes, 374
communications backup, 67, 78, 80, 198, 289
communications connectivity, 325, 445
communications protocols, 4, 9, 17, 24, 30, 36, 53-55, 64, 87, 364, 381, 385, 455, 465
communications security, 77. *See also* encrypting modems; VPN technologies
across analog circuits, 75, 78, 371
IP/SEC, 91, 96, 457
IPv6 (IP version 6), 84-85, 88, 91, 93, 95-96, 177, 179-180, 184, 196, 254, 359, 457, 473
link encryption, 51-52, 66, 175, 239, 369, 381, 410, 424, 440
maintenance of, 395, 417
in SCADA systems, 424
VPN technologies for, 75
compact discs (CDs), 227, 299, 300. *See also* removable storage media
COM ports, 4-5, 65, 346, 470
computer hardware, 49, 100, 124, 212
computer storage limits, 157-160, 190
computer systems, xvii, xxi, xxii, 16, 46, 76, 84, 86-87, 125, 131, 171, 176-178, 186, 188, 208, 211, 231, 234, 242, 430, 434, 441, 447, 454-455, 473
concentrators, 47, 366
confidentiality, 52, 76, 80, 84, 91, 170-171, 214, 218, 427-428, 436-439, 468
confidential material management, 298, 303
confidential materials, 288-289
configurable utilities, 116
configuration changes, 113, 116, 212, 249, 290, 335, 338, 382, 394, 418, 429
configuration data, 35, 99, 110, 155, 219, 271, 333, 335, 443
configuration maintenance utilities, 105, 107
configuration tables, 104, 109, 112, 170
configuration utilities, 103, 110, 212, 382
connection-oriented communications, 72, 74
connection-oriented telephone circuits, 72
connectivity, types of, 18-19, 58, 70, 72-75, 78-80, 84-85, 95, 131, 152, 159, 172-174, 194, 224, 238, 241, 249, 252, 276, 278, 325, 333, 342, 360, 368, 370-371, 377, 408, 438, 445, 449, 470
console port, 31, 34, 38, 41, 48-49, 205, 293, 320-321, 346, 447
contact bounce, 26
contact inputs, 26, 28, 45, 115, 149-150, 295, 364
contact outputs, 23-24, 27-29, 37, 112-113, 118, 364, 366
contractors, 116, 178, 230-231, 301, 309, 412, 416, 434
control algorithms, 446
control interfaces to plant system controls, 387
control-loop cascades, 447
control-point tagging display, 156
control room console design, 130
conventional information technology security
availability, integrity and confidentiality, 52, 423
firewalls, 321-322
intrusion detection systems, 328
SCADA architecture, 252, 305, 324, 423
wireless local area networks (LANs), 344, 386
co-ops. *See* rural electric cooperatives (co-ops or RECs)
Corel Draw, 138
corporate culture, 297
corporate LANs/WLANs
authentication/validation process on, 293
data integration with, 390
gateway computers on, 180
malware (malicious software) on, 186, 193
remote access to, 93, 172, 177, 179, 209
SQL injection against, 124, 266, 271-273, 406, 469
substation connections to, 372-373, 471
virus scanning on, 435
countermeasure selection, xvii, 171, 192, 237-240, 244, 248-249, 278, 301-302, 311-312, 372, 417, 422
counterrotating ring design, 82-83
covert channels, 267, 277, 313
cracker, 172, 260, 368, 447
crackers, 467

credentials, 203-204, 207, 210, 283-285, 288, 291-292, 301, 303-304, 309, 428, 430, 434, 440-441, 444-445, 450, 463
creeping integration, 396
critical ancillary subsystems. *See also* alternative (backup) facilities
  electric power, 293-294
  fire suppression, 291
  local area networks (LANs), 293
  telecommunications, 292
critical software modules, 240-241, 248, 298, 416-417
critical utility authentication/validation process, 172, 204, 208
current-alarm summary display window, 151
current state, 1, 5, 149
customer training, 458
cyber attacks and cyber threats. *See also* attacks; insider attacks; malware (malicious software); social engineering; vulnerability points
  means of attack, 231
cybersecurity issues
  electric utility industry-specific, 299
  in SCADA systems, 94, 151
cybersecurity tests, 256-257, 266, 298, 301, 307-308, 384, 417
cybersecurity vulnerabilities
  attack and vulnerability points, 321, 407
  historian/trending package, 345
  impact of successful attack, 229
  most frequent means of attack, 369
  probability of attack, 229, 238
  risk assessment, 297, 397, 399, 410, 416
  threat and threat agents, 458
cyclic redundancy check (CDC) code, 50
cyclic redundancy check (CRC) code, 53, 170, 220-221, 448

D

daemon dialers, 173, 448, 473
dangerous practices, 102, 134, 148, 198, 209, 230, 243, 280, 282
DARPANET, 178
data and control exchange, 109, 122, 126, 373, 419
database point utility, 111, 113, 158, 165, 448
data encrypted standard (DES), 215
data-exchange protocols, 464
data-exchange standards, 122
datagram, 85, 87-88, 90, 95, 178-180, 185-189, 195-196, 249, 315-316, 318-319, 329, 443, 448, 453, 472
datagram header information, 185
data link connection interface (DLCI), 81, 450
data sampling, 159, 339
data storage hierarchy, 159
data transmission rates, 3, 64, 73
daylight savings time, 46
day zero, 449
DCOM. *See* distributed common object model (DCOM)
dead man timers, 9, 449
debouncing, 27
decluttering, 146-147
deep-inspection firewalls, 313, 316, 449, 469
Defense Department Advanced Research Projects Agency, 84, 178
defense-in-depth security strategy, 396
demarcation point, 64
denial of service (DoS) attacks, 188, 191, 276, 451, 475
Department of Defense Advanced Research Projects Agency (DARPA), 84
deterministic networks, 82-83, 96, 468
diagnostic displays, 132-135, 144
dial-back modems, 438
dial-in relay connectivity, 368
dial-in telephone lines, 369
dictionaries, 339
digital cameras. *See* mobile electronics
digital-certificate-based authentication, 91
digital certificates, 91-93, 132, 207, 210, 221-223, 226, 303, 361, 371-372, 430, 435, 444-445, 450, 463, 474
digital circuits, 56, 64, 71, 292, 371
digital circuits vs. analog circuits, 64, 71
Digital Equipment Alpha computers, 85
digital networking technologies, 65, 80, 371
digital radio, 67, 379
digital signal processors (DSPs), 46
digital signatures, 221, 223, 450
digital subscriber line (DSL) circuit, 71, 451
digital telephone company, 71
digital-to-analog (D2A) converters, 70, 450

digital versatile discs (DVDs), 157, 160, 232, 286, 291, 299, 308. *See also* removable storage media
digitized signals, 70, 78, 451, 473
direct access, 177, 447
direct current (DC), 43, 294
directed attack probabilities, 398
directional (yagi) antennas, 380
direct memory access (DMA), 10, 450
direct-sequence spread spectrum, 76, 451
discrete Fourier transform (DFT), 46
disgruntled employees, 230, 232
display hierarchy, 144-145, 450, 461
display navigation, 145, 147, 450
distributed agents, 459
distributed architectures, 12-13, 102
distributed common object model (DCOM), 14, 122, 125, 255, 349, 351, 447, 449, 461
distributed control system (DCS), xxii, 39-41, 50, 106, 124, 198, 206, 230, 240, 295, 343, 364, 372, 382, 397, 400, 415, 421, 449, 453, 455, 463-464, 470
distributed denial of service (DDoS) attacks, 188, 191, 276, 449, 451, 475
distributed operator-console architecture, 40
distributed supervisory control and data acquisition (SCADA) system architecture, 13
distributed Web-based HMI, architectures, 12
distribution systems, 271
DMZ, 124, 226, 252, 312, 343-345, 378
DNP3.0, 36, 80, 373-374, 451
DNS cache, 89, 358, 451
dongle, 131
double duty computers, 218
doubly encrypted scheme, 449
downloaded calculation functions, 10, 38, 393
downloaded logic and parameters, 10, 30, 36-38, 382
downloading into remote terminal units (RTUs), 10, 30, 36-38, 41, 49, 107, 110-111
drill-down capability, 142-143, 351
drivers, 31, 55, 350
DROP (command), 272, 469
DS0 circuits, 62, 296, 451
DSL technologies, 75, 371
dual-homed PCs, 396
dual-key (public key) encryption, 215, 217, 221, 463, 465
dual-ported capability, 11
dual-use PC bridging, 360
dumb remote terminal units (RTUs), 8-9, 26-29, 36, 44, 47, 52, 74, 385
dumb terminals, 69, 74, 474-475
Dupont Network Security Assessment Methodology (DNSAM), 248
DVDs (digital versatile discs), 157, 160, 172, 228, 232, 286, 290, 291, 299, 308, 350. *See also* removable storage media
dynamic host configuration protocol (DHCP), 182, 200, 266, 351, 449

E

electrical substations. *See* substations
electric power
  battery backup, 295, 470
  generator ramping, 366, 454
  standby generator, 294
  subsystem restoration procedures, 300, 303
electric utility industry. *See also* substations
  NERC 1200/1300 compliance, 299, 415
  SCADA system development, 1, 130
electronic access controls, 283-285, 288-289, 417, 443
electronic communication paths, 84, 172-173, 184, 324, 369, 472
electronic perimeter, 311, 416, 436-437
electronic security incident response and reporting procedures, 302
electronics system security
  Bluetooth-enabled devices and, 76, 194, 461
  computer systems and, 301, 340, 387, 415, 427
  mobile electronics and, 268, 273-275, 277-279, 298, 355-356
  PCs and, 311, 330, 332, 336, 347, 354, 356, 360-361, 364, 380, 394, 396, 412, 435-437
  removable storage media and, 172, 212, 219, 298, 430, 441
  RTUs and, 305, 364-366, 368-369, 372-373, 377, 379-382, 389, 393-395, 401, 423-424

electronic vaulting, 452
e-mail, 85, 88, 152, 191, 193, 209, 214, 218, 251, 273, 293, 296, 298, 304, 312, 319-320, 331-332, 341, 343, 345, 357-358, 360, 378, 395-396, 400, 428, 433, 441, 472, 474
e-mail attacks, 89, 183, 233-235, 242, 268, 273, 276-277, 469
e-mail clients, 315, 451
e-mail integration, 396
emergency procedures, 300
emergency shut down and start-up logic, 384
emoticon, 137
employees. *See also* insider attacks; social engineering
  background checks of, 239, 301, 309, 428, 434
  disgruntled, 230, 232
  education of, 212, 235, 301, 306-308, 328, 439, 458
  ex-employees, 105, 231-232, 306
  financial checks on, 230, 309
  former insiders, 231, 251
  hiring policies and procedures, 230, 285, 298, 412-413
  initiation and termination procedures, 298, 301, 428
  security training procedures, 302, 306, 434
  training of, 48, 105, 153, 213, 290, 298, 302, 306-309, 328, 412-413, 416, 434, 439, 444, 458
encapsulated protocols, 57, 91
encapsulation, 57, 452
encapsulation devices, 91
encrypting modems, 438
encryption
  to protect transmitted information, 51-52, 66-67, 78, 91-92, 136, 171, 207, 214-216, 218-221, 233, 266-267, 277, 335, 350, 354-355, 368, 372, 374, 384-385, 413, 417, 421, 424, 440, 446-447, 449, 452, 457-458, 463-464, 467-470, 473
  for sensitive information, 91, 233, 413
encryption and ciphers
  hash code, 220-221
  key size, 217
  shared secrets, 216
  VPN technologies, 218
encryption keys, 52, 91-92, 207, 215, 219, 233, 450, 452, 458, 465, 467, 473
encryption technology, 219
end-to-end authentication, 91
end-to-end VPN, 93, 355
energy management systems (EMS), 145
engineering unit (EGU) conversion factors, 36, 110, 452
engineering unit values, 35-36, 38, 110, 452
engineers, 105, 206, 212, 369, 425
environmental factors, 291
e-terrorism, 188, 214, 229, 231-232, 236, 251
Ethernet, 14, 54, 59, 65, 72, 76, 78-79, 84-87, 122, 135, 178, 184, 186, 192-193, 196-198, 200, 293, 313, 318, 326-327, 333, 335-336, 343, 347, 356-357, 360, 367, 384, 423
Ethernet DNP, 367, 372-373
Ethernet local area network (LAN), 15, 17, 55, 57, 82, 86, 88, 90, 97, 127, 169, 174, 181, 192, 194-195, 197-200, 285, 293, 346, 372-373, 383, 452, 458, 471, 474
Ethernet Modbus, 313, 322, 367, 372-373
Ethernet ports, 31, 34, 172, 287, 382
Ethernet switch configurations, 185, 195, 293, 316, 318, 321, 346, 391, 393, 436, 443
Ethernet technologies, 458
ethical hackers, 172, 452, 454
evil-twin AP, 452
ex-employees, 231
exploitation mechanisms, 190-191, 236, 244
exploring, 189, 265
extensible markup language (XML), 58, 125, 271, 444, 474
extensible markup language (XML) data files, 59, 271
external hard drives. *See* mobile electronics
external link encryption devices, 424
external threats, 230-231, 309
extranet, 178, 452

F

false negatives, 224-225, 284, 333, 341, 452
false positives, 224-225, 266, 284, 328, 333, 341, 453
fast Fourier transform (FFT), 46
fault tolerant systems, 14, 17, 40, 82, 170, 236

Federal Communications Commission (FCC), 66, 378, 380
fiber-distributed data interface (FDDI), 72, 82, 84, 179, 181, 386
fiber-distributed data interface (FDDI) counterrotating ring design, 82-83
fiber-optic cables
availability and uses of, 370
bandwidth capacity of, 69, 370
development of, 370, 393
evolution of, 68
of munies, 370
field sites, 5, 16, 42, 50, 57, 61, 70, 73, 75, 133, 159, 172, 174-175, 207, 238, 294-296, 322, 325-326, 342, 382, 384, 457, 459
file access controls, 140, 218
file encryption, 219, 228
file protection mechanism, 219
file transfer protocol (FTP), 58, 225, 269, 454
financial checks on employees, 230, 309
fingerprint scanning, 210, 282
fire alarms, 282, 304, 440
fire suppression, 291
firewalls, 186, 189, 202, 337, 356. *See also* corporate LANs/WLANs; TCP/IP (transmission control protocol/Internet protocol); Web site/server
configuration of, 186, 191, 318, 320, 322-323, 350, 436, 440, 442
covert channels, 277, 313
deep inspection, 313, 316, 449, 453, 469
deep-inspection, 316
vs. denial of service (DoS) attacks, 188, 191
described, 185-186
and DMZ, 344, 378
dual, 193-194
and ICCP, 373
inspection, 186, 191
limits to, 186
and N/HIDS (network/hostbased intrusion detection systems), 224, 251, 267, 302, 304, 322, 332-333
placement and uses of, 186
procedures regarding, 191, 251, 302, 304, 333
protection from probing by, 189
responsibility and maintenance of, 313, 442
for SCADA/LAN interconnections, 192, 313-316
vs. scripting, 275
vs. SYN flooding attacks, 190, 469
Trojan risk, 314, 400
updating procedures, 200, 437
and virus scanners, 224, 304, 314, 442, 472
and VPN, 186, 313-315, 320, 346, 355-356, 372
vulnerabilities of, 189, 239, 249, 251
Web site/server application proxy, 315, 449, 453
vs. zombie computers, 191
firmware, 30, 36-37, 107, 220-221, 228, 266, 453
firmware changes, 30
first out alarm, 150
the Fix (commercial software), 14, 122
flags, 32, 37, 126, 148, 384, 463
flash cards. *See* removable storage media
flash drives, 172. *See also* removable storage media
floppy disks. *See* removable storage media
former employees, 213, 230
former insider, 231, 251
frame relay
described, 57, 74, 81
digital features of, 58, 75, 81
used for replacement of SCADA analog leased lines, 72, 74-75
frame-relay access devices (FRADS), 74-75, 431
frame relay DCLI-to-IP address mapping, 81
frame-relay digital communication, 75
frame-relay networking and virtual circuits, 75, 81, 450, 465, 472
frequency bands, 378-380, 453, 472
frequency hopping, 76-77, 380, 453-454
front-end computers, 12
full-duplex channels, 444, 454

G

gas pipelines. *See also* pipeline industry
nature of, xvii, 1, 389, 458
remote terminal units (RTUs) used in, 30
RTU backup power supply from, 43, 147

gateway computer, 180, 183
gateways and IP addresses, 95, 179, 181
generalized software architecture, 20
generator ramping, 366, 454
geographical layout operational display, 142
geographic information system (GIS), 142, 145-147
geographic information system (GIS) maps, 141, 144-145
global positioning system (GPS)
  satellite network, 45, 147, 454
  time receivers, 342, 365
GOOSE messages, 367, 373
graceful degradation, xxi, 14, 236, 454
GRANT (command), 469
graphical display editor, 106, 113-114, 145
graphical displays, 17, 113, 115, 136, 139-143, 145, 147
graphical editors, 113, 138
graphical presentation formats, 143
Greenwich mean time (GMT), 46
GSM technology, 80

H

hacker attacks, 445
hackers. *See also* cyber attacks and cyber threats
  ethical, 172, 452
  getting through firewalls, 186, 188, 313
  and password systems, 105, 219
  remote access for, 172-174, 176-177, 400
  white-hat, 172, 452
hacker tools, 86, 176, 191, 202, 255, 448, 454, 462-463, 467, 474
hacker Web sites, 173, 183, 255, 268, 454, 473
half-duplex channels, 64, 455
Halon systems, 291-292
hardware security chip, 220
hash algorithms, 220-221, 448
hash code, 117, 220-221, 455
hash messages, 221
high-level checks, 439
high level languages, 118, 447
hijacking, 380, 392, 424
HIPPA, 289, 299, 413, 428, 440, 455
hiring policies and procedures, 230, 285, 298, 412-413
historical trending, 19, 137, 156-162, 166
historical trending displays, 161-162, 211
HMI (human-machine interface), 2, 6, 16, 112, 115, 144, 146, 209, 271, 400, 455, 468
host address, 182
host authentication, 175
host-based intrusion detection systems (HIDS). *See* N/HIDS (network/host-based intrusion detection systems)
host computer, 1, 3, 6, 8-10, 26, 32, 35-37, 41, 44, 51, 53-54, 58-59, 65, 70-71, 75, 94, 114, 176, 325, 393, 459
hot spots, 298, 455
hot stand-by schemes, 12
HTML injection, 275
HTTP/S, 58, 91, 175, 210, 218, 254, 275, 400, 410, 448, 455, 474
human-machine interface (HMI), 2, 6, 16, 112, 115, 144, 146, 209, 271, 400, 455, 468
hybrid access controls, 282, 284
hybrid access control system architecture, 282
hybrid delivery mechanisms, 194
hypertext transfer protocols (HTTP and HTTPS), 58, 91, 175, 210, 218, 254, 275, 400, 410, 448, 455, 474

I

ICCP protocols, 54, 58, 80, 126-127, 364-365, 372-374, 455
identification number (ID) of remote terminal units (RTUs), 5
ID/password scheme, 130-131
  described, 130
  linked to job-category-based user accounts, 130-131
  for remote terminal units (RTUs), 131
  security level through, 131
  storage of, 131
  weakness with, 130-131
IEC61131-3 standards, 49-50, 467
IEEE 802.11 wireless Ethernet, 174, 207, 346, 378
IEEE 802.11x, 174, 207, 346, 378
IEEE 802.16 wireless Ethernet, 78, 378
IETF standard, 315
illegal-entry alarms, 288

impact of successful attack, xxi, 398
in alarm (condition), 152
incident response and reporting procedures, 236, 280, 302, 304, 338, 398-399, 411, 417
independent system operator (ISO), 457
infinite loop, 11, 456, 474
information collecting malware, 191, 277
information flow within SCADA systems, 20-21
information technology (IT) systems, 14, 73, 169
infrastructure mode, 460
inside knowledge, 171, 212, 214, 230, 393
insider attacks, 251, 279, 286, 288, 306
insiders as threat agents, 230-231, 423
instant messaging (IM), 90
integrated service digital network (ISDN), 72-73, 448, 457
integration, 23, 67, 125, 212, 308, 390
integrity, 52, 54, 76, 91, 111, 115, 117-118, 128, 136, 161, 170-171, 214, 221, 280, 331, 423, 448, 468
integrity poll, 53, 148
Intel-compatible microprocessors, 17
intelligent electronic devices (IEDs), 33, 369, 372-374
intercommunication, 34
interconnection points (inter-ties), 366
inter control center protocol (ICCP). *See* ICCP protocols
interdisplay navigation and display hierarchy, 145, 450
internal threats, 230, 279
International Standards Organization—Open Systems Interconnect (OSI), 178
International Telecommunications Union (ITU) X.509 format, 222-223, 444-445, 450, 474
Internet
  described, 17, 61, 76, 84-85, 95
  origins of, 74, 84
  privatization of, 84
Internet control message protocol (ICMP), 189, 316, 320, 322, 352, 455
Internet Engineering Task Force (IETF), 315
Internet protocol (IP), 57, 87, 126
Internet service providers (ISPs), 95
inter-networking protocol (IP), 85
interoperability, 8, 17, 54, 177-178, 456
interpreted languages, 49, 456
interpreter programs, 456
intersystem and intrasystem data exchanges, 17, 177
intersystem data exchange approaches, 17
intranet, 178, 456
intrusion detection devices (IDSs), 104
intrusion detection systems, 286. *See also* N/HIDS (network/host-based intrusion detection systems)
intrusion-prevention systems, 104, 118, 286, 297, 326, 410, 455
IP addresses, 72, 88-90, 103, 175, 177, 179-180, 182, 185-186, 196, 200, 209, 233, 257, 318, 320, 321, 328, 331, 351, 358, 384, 444, 451, 457, 460, 463
IP address scanner, 463
IP address sweeping, 176
IP along the pipeline, 395
IP-based communications, 163, 294, 423
IP-DNP3.0 protocols, 80
IP-DNP protocols, 80
IP link encryption, 175, 239, 381
IP messages, 57, 78, 82, 90, 95, 105, 185, 452, 460, 463, 465
IP-Modbus protocols, 80
IP network architecture, 86
IP networking
  in the field, 296, 395
  at substations, 365, 372
IP-ready remote terminal units (RTUs), 57-59, 78, 424
IP SCADA protocols, 80
IP security IP(SEC), 374, 388
IP spoofing, 186, 443, 457
IP suite (of protocols), 84, 89, 177-179, 189, 449, 454-455, 470
IP to the field, 172, 176, 322, 325, 342, 355, 385, 424, 457
IP to the substations, 365, 371
IPv4 (IP version 4), 84-85, 89, 91, 95-97, 176-177, 179-180, 182, 184-188, 195-196, 359, 457, 473
IPv6 (IP version 6), 84-85, 88, 91, 93, 95-96, 177, 179-180, 184, 196, 254, 359, 457, 473
IP wide-area network (WAN), 85
islanding, 366
ISO/OSI model, 55, 318, 457, 464
IT groups, 169, 305

J

Java applets, 119, 457
JavaScript, 104, 118, 273-275, 305, 456, 467
job category, 106

K

Kerberos technologies, 242
key, 217-219, 221-222, 233, 281-283
key creation, 174
key exchange, 174, 457
key size, 207, 217

L

laptop PCs, 34, 43, 48, 220, 231, 233, 239, 278, 286, 347, 354, 356, 380. *See also* mobile electronics
layering, 99, 123, 147
layers of defense, 289, 411
leased analog telephone circuits, 23, 379
leased T1 circuits, 70
leased telephone lines, 63-64, 67-68, 74, 292, 295, 364, 377, 379, 386, 437-438
least privilege concept, 279
legacy protocols, 4, 10, 50-51, 54, 363-364, 374
licensed radio, 3, 61, 65-66, 379-380
licensed radio communications, 379-380
link encryption, 51-52, 66, 126, 175, 239, 369, 381, 410, 424
link-layer encryption, 80
liquid pipelines, xxii, 389-390, 393, 458
local area networks (LANs). *See also* corporate LANs/WLANs
  bridging of, 193, 444
  as critical ancillary subsystems, 291
  Ethernet, 15, 17, 55, 57, 82, 85, 88, 90, 97, 122, 127, 169, 181, 192, 194-200, 202, 285, 373, 444, 446, 452-453, 458, 471
  origins of, 12
  telephone circuits used to bridge, 73
  virtual, 201, 312, 356-357, 472
  wireless, 204, 206-208, 445, 462, 474
local clocks, 44
local display, 34, 36
local loop, 63
local regulatory control panels, 39
lockout, 351
login, 106, 130-131, 191, 207, 293, 339
login attempt restrictions, 209, 331, 351, 445, 462
logs, 102, 109, 118, 153, 205, 212, 266, 288, 303, 331, 337-343, 347, 353, 429, 441
logs and reports, 6, 20, 36, 164
long-distance communications, 61, 85
longitudinally redundant check (LRC), 53
low-level checks, 149
low-power operation, 42-43, 389

M

Macromedia Flash, 160, 241-242, 403
magnetic tape cartridges, 160, 308
malformed messages, 189
malicious supervisory applications, 395
malware (malicious software)
  defined, 191
  delivery of, 191, 193-194, 228, 286
  detection of, 186, 224-225, 313, 324, 328, 330, 333
  installation of, 286
  and N/HIDS (network/hostbased intrusion detection systems), 176
  sent from Web/site server, 191, 234
  types of, 114
  worms as, xxi
malware scanning, 228, 312
management utilities, 103, 107
man-in-the-middle attack, 200, 266, 455
man-machine interface (MMI), 1, 459
manual access controls, 282-283
manual mode control loops, 447
manufacturing messaging service (MMS), 127, 373
map board, 6, 23, 459
mass storage, 158
master and slave protocols, 32-33
master terminal units (MTUs), 23, 459
means of attack, 273
measurement conditions, 154
media access control (MAC) addresses, 87, 90, 195, 197-198, 200-202, 204, 209, 318, 320, 346-347, 444, 452, 458

media players. *See* mobile electronics
message authentication code (MAC), 87, 90, 458
message-by-message encryption, 91
message digest, 221, 355
message packet encryption, 70, 224, 457, 472
message payload, 87, 459
metallic circuit, 63, 459, 470
microcomputers, 172, 220, 223, 228, 231, 233, 239, 278, 380, 400, 473
microprocessors, 3-4, 8-10, 14, 17, 19, 29, 67, 270, 463, 466, 468
Microsoft PowerPoint, 138
Microsoft Windows/Intel (Wintel) platform, 121, 123
Microsoft Windows operating system, 122, 192, 447
microwave based private telephone systems, 68
microwave relay towers, 3, 391
microwave repeater stations, 65, 68, 393
mimic panel, 6, 23, 144, 459
minicomputers, 8, 459
mitigation and risk assessment, 237-238, 242, 244
mobile ad-hoc networking, 445
mobile code, 273-275, 458
mobile electronics, 5-6, 20, 78, 172, 349-350, 360, 394, 409, 424, 448, 458
Modbus protocol, 37, 49-50, 381
modems, 3, 63, 64, 73, 75, 295, 378. *See also* encrypting modems
  analog, 173, 295
  auto-answer, 173
  cellular, 79, 378
  dial-back, 438
Morse code, 61-62
MTU (computer), 6, 23-24, 459
multicast/multimedia backbone (MBONE), 95
multi-dropping, 6, 53
multi-factor authentication, 132, 209-210, 417, 460
multiple address system (MAS), 67
multiple master radios, 65-66
multiplexed communications towers, 65
multiplexing analog inputs, 25
municipal area networks (MANs), 386, 459
municipal LANs and wide-area networks (WANs), 78, 378, 386
municipal SCADA systems, 79, 364, 370, 377, 387
municipal utilities (munies), 373
  fiber optic network wide-area networks (WANs) and UCA2, 370, 374, 386
  vulnerability points of, 370

N

National Institute of Standards and Technology (NIST), 374, 406, 425
natural gas liquids (NGL), 389
NERC 1200/1300 compliance, 374-375, 460
NERC recommendations, 299
nessus (software package), 249
network addresses, 95, 180
network address translation (NAT), 95, 180
network-based serial protocol architecture, 56
networked demilitarized zone (DMZ), 124, 226, 252, 312, 343-345, 378
network interface cards (NICs), 90, 193, 197, 360
network mappers, 460, 470
network protocols vs. serial protocols, 31, 55-56
N/HIDS (network/host-based intrusion detection systems), 104, 224, 251, 267, 287, 302, 304, 311, 322, 330-333, 337, 341, 345, 350, 410-411, 442, 455, 460
  described, 176, 326, 460
  and insider activity, 287
  and malware, 94, 176, 186, 251, 328, 330, 333, 355
  operator tracking, 287
  updating procedures, 301-302
  and worms, 251
nmap, 189, 258-259, 265, 470, 475
nonrepudiation, 186
nonsecure protocols, 381
nonspecific attack probabilities, 232

O

object linking and embedding (OLE), 14, 461
object linking and embedding for process control (OPC) standards, 14-15, 17, 90, 117, 121-122, 313, 322, 351, 372, 461

off-line configuration changes, 301, 305
off-line storage, 160, 460
offshore (oil/gas) production, 78, 143
off-site storage
    procedures for, 290-291, 303
    of system backup materials, 290-291, 303, 441
oil pipelines. *See also* pipeline industry
    nature of, 26, 389, 400
    remote terminal units (RTUs) used in, 365, 389, 393
omnidirectional antennas, 67, 380
one-way algorithms, 220
online configuration changes, 204-205, 382
online storage, 160, 460-461
on-site storage, 441
OPC
    described, 14-15, 122, 461
    third party software and, 14, 117, 122
    uses of, 122, 127, 163
    vulnerabilities of, 123, 126
OPC client/server types, 122
OPC configuration alternatives, 122
OPC data interchange standard, 122, 372
OPC Foundation, 123
OPC servers and client/server relationship, 122
open database connectivity (ODBC), 166
operating systems. *See also* Microsoft Windows operating system; Unix/Linux operating system
    access controls to, 105, 115, 118, 130, 210
    administrative programs of, 103
    deficiencies of, 100-101
    passwords for, 130-131, 298
    protection mechanisms of, 124, 153, 197, 313, 329-331, 335, 339, 346, 445
    for SCADA systems, 14, 20, 99, 105, 115, 121, 132, 211, 410
    versions of, 8, 17, 56-57, 59, 254, 358, 361
    vulnerabilities of, 100-102, 190, 240, 349, 403
operational access, 105, 140, 388
operational differences, 105, 304-305
operational security
    procedures, 240
operator, 1, 3, 5-10, 13-14, 17, 23, 30, 36-37, 62, 102, 104-106, 108-109, 114, 116, 120-121, 133, 136, 139-144, 146-148, 150-151, 153-155, 163, 167, 192, 213, 302, 306, 308, 360, 384, 388, 394-395, 443-444, 449, 457, 470
operator authorization, 395
operator-console architecture, 40, 99, 106-107, 118, 130, 144, 211, 388, 396, 461
operator controls, 140, 153
operator interface
    standards system displays, 212
operator permission, 121, 155
operators, 20, 46, 58, 64-66, 68, 99, 112-113, 119, 129, 131-132, 138, 143-144, 147-149, 211, 296, 305, 384, 391, 400, 445
operator tracking, 105, 153
out-of-band messaging, 468
out-of-band signaling, 67, 461
overloading, 451
overview display, 143, 450, 461

P

packet, 189, 312-313, 316, 318, 324, 329, 448, 461, 471, 473
packet assembler/disassembler, 70
packet filtering, 88, 90, 94, 185-186, 315-316, 318, 324, 453
packet sniffers, 423, 462
PAD, 70, 74
pager based notification, 152
pager systems, 152
panel instrumentation, 40-41
panning, xvii, 112, 145, 147, 399, 459
pass-through mode, 369, 462
password authentication protocol (PAP), 131, 208-209, 347
password-crackers, 267, 462
passwords. *See also* ID/password scheme
    expiration of, 301, 439
    procedures for, 130, 208-209, 298
    strategies for, 132, 204, 207, 210, 293, 301, 350, 354
payload, 188, 241-242, 249, 252-255, 269-270, 275, 277, 315, 340, 461
PCMIA cards, 172, 349-350, 360, 409
PCMIA plug-in storage devices. *See* mobile electronics
PCs, 14, 17, 34, 43, 48, 73, 85, 102, 134, 192, 220, 231-233, 239, 278, 286, 311, 330,

332-333, 336, 347, 354, 356, 360-361, 364, 380, 394, 396, 412, 435-437
peer-to-peer communications, 58
penetration test (pen test), 256, 265, 307
peripheral transfer switching, 14
permanent virtual circuits (PVCs), 74-75, 465, 472
personal area networks (PANs), 462
personal computers, 14, 48, 441
personnel system access, 106, 115, 129-131, 139, 173, 209, 211-213
physical access rights, by job category, 230-231, 239, 279, 292
physical isolation of assets, 289
physical protection of materials and information, 289-294, 303
physical security
  access controls, 235, 240, 326, 386, 417, 419, 440, 443
  field site, 324, 384
  incident response and reporting procedures, 302
  physical isolation of assets, 419
  physical protection of materials and information, 326, 375, 392-393, 417-418, 440
  remote sites, 410
physical security layers, 281, 289
ping sweep, 176, 463
PINs, 210, 219, 235, 240, 281, 283-284, 298
pipeline industry
  IP along the pipeline, 70
  radio communications, 69-70, 84
  RTU program logic, 36, 38-40
  SCADA systems architecture, 305, 377
  smart remote terminal units (RTUs), 30
  specific cybersecurity issues, 400
  supervisory control applications for, 236
plaintext, 214-215, 446, 449, 458, 463
plant a flag messages, 32
plant DCS and SCADA system, 343, 364
point group display (bar graph mode), 136, 138
point-to-multipoint ultrahigh-frequency (UHF) radio transmissions, 65-67, 76-77, 379, 472
point-to-point high-bandwidth microwave transmissions, 65
point-to-point protocol (PPP), 56
point-to-point tunneling protocol, 355, 473
poke points, 119, 142, 144, 450, 461
policies and administrative controls, 297
policies vs. procedures, 304
polled communication mode, 374
polled report by exception communication mode, 53, 374, 451
polling, 2, 4, 10, 12, 30, 32, 38, 43-44, 47, 49, 51-54, 56, 58, 65, 67-68, 78, 99, 107, 109, 132-133, 159, 163, 174-175, 214, 238, 302, 365, 369, 379-380, 382, 384, 424, 438, 448-449, 456-457, 463, 465, 472
polling cycles, 5, 23, 31, 148, 377
ports, 4-5, 12, 31-33, 59, 65, 74, 88-90, 103, 172, 175, 188-189, 197-199, 201-204, 249, 287, 293, 320, 322, 326-328, 331, 347, 351-352, 357, 360-361, 368-369, 382, 409, 417, 443, 451, 453, 463-464, 470
port scanners, 463
port sweeper utility (nmap), 189
port switches, 368-369, 463
power supply. *See also* electric power; substations
  battery backup, 43
  configuration and equipment interconnections, 294
  solar power, 42-43, 468
  standby generator, 294
  thermoelectric generators, 42-43
  uninterruptible power supply (UPS), 170
primary unit, 10
printed logs, 153
private branch exchange (PBX), 63
private key, 91, 93, 217-219, 221-222, 463-465
private telephone systems, 68-69
probability of attack
  directed attack probabilities, 229, 238
probing and exploring, 189
procedures
  critical sets of, 290, 301-304, 308-309
  vs. policies, 304
  validation of, 301, 303-304, 308
procedures and policies, 170, 213, 235, 248, 251, 283, 299-300, 305-306, 308-309, 427, 468
processes, xxi, xxii, 1, 6, 23-24, 53, 68, 87, 112, 162, 164, 170-171, 227, 236, 270, 289, 297,

302, 305, 311, 323-324, 338, 377, 411-413, 417-418, 423, 464, 467
process-flow diagrams, 138, 143
programmable logic controllers (PLCs)
emergence of, xxii, 49-50
programming and configuration downloading evolution, 382-383, 463
remote changing of logic and configuration data, 155, 333, 382
as remote terminal units (RTUs), 19, 25, 36-37, 41-42, 49, 58, 133, 381-382, 394, 463
programmers, 118, 121, 212, 270, 330
programmer-specified trigger malware, 118, 270
proportional-integral-derivative (PID) controller, 39-40
protection mechanisms of operating systems, 104, 115, 130-131, 153, 191, 218-219, 225, 240, 254-255, 270, 329, 445
protocol analyzers, 374, 380, 385, 464
protocol converters, 373, 464
protocol standards, 49, 54, 382
protocol test set software, 54, 380
proxy servers, 313, 315-316, 453, 464
pseudo-points, 110
public key encryption, 215, 217, 221, 463
public-key infrastructure (PKI), 463
publish-and-subscribe mechanisms, 125
pulse inputs, 28, 43
pulse outputs, 28

Q

quality codes, 161
quality of service (QoS) requests, 96

R

radio communications, 3, 54, 62, 65, 67, 77-78, 292, 379-380, 391-392, 472
radio frequency ID (RFID) smart card, 284, 465
radio spectrum, 380
radio systems, 67, 76, 377, 379
RADIUS (software packages), 204, 337
raw counts, 35-36
real time
defined by usage, 1, 56, 62, 78, 95-96, 121, 130, 140, 164, 176, 329, 337, 355, 363-364, 367, 419, 443, 458, 465
and telecommunications technologies, 58, 75, 94, 96, 132, 163
real-time communications connections, 364
real-time data
and command exchanges, 163, 166, 372, 395
and computer development, 122, 163
ICCP for, 372
Internet developments towards, 58, 96
for pipelines, 390
vs. real-time trending, 158
SCADA requirements for, 170
SCADA systems and, 2, 148, 156, 271, 464
from substations, 366, 369
through DSL, 75
uses of, 58
Web server functions and, 271
real-time trending vs. real-time data, 162-163
recovery procedures, 251-252, 307-308, 398
redundancy schemes, xxi, 10-12, 14, 41
redundant computer, 10
redundant operational systems, 302
reed relays, 25
regulatory and sequence control, 36, 39-41, 50, 382, 390, 393
relational databases
SCADA system uses of, 145, 170, 271
SQL commands to, 272
SQL-compliant, 272
vulnerabilities of, 272
and Web pages, logical relationship between, 271
relational database server, 123
relay ladder logic (RLL), 50
remote access. *See also* dial-in telephone lines
to SCADA systems, 173, 177, 179, 209, 238, 470
VPN configuration for, 131, 223, 388, 428, 435
remote access Trojan (RAT), 276
remote backup control center. *See* alternative (backup) facilities

remote console support, 3, 58, 61, 129, 209, 384, 387, 401, 418
remote control access, 209-210, 395
remote electronic access procedures, 194, 303
remote intelligent gateways, 79, 179
remote login attempts, 331, 351, 462
remotely located master radios, 65-66, 379
remote party entrance malware, 355
remote procedure calls (RPCs), 14, 122, 461
remote sites, 68, 382, 410
remote terminal units (RTUs)
authentication and message encryption for communications, 67
contact output (control) type, 113, 118, 175
current value display, 136
defined, 466
downloading into, 107
electronics system security and, 19, 294, 369
microprocessor-based, 393, 445, 455, 466
in pipeline applications, 377, 389, 393-395
polling and communications diagnostic display, 132-133, 463
polling channel status display of, 132
programmable logic controllers (PLCs) as replacement for, 381-382, 463
protocol with adjustable supervisory points, 118
vulnerability points of, 238
removable media connection points, 160
removable storage media, 227, 283
electronics system security and, 160, 172
policies regarding, 303
SCADA data storage on, 160, 361
sneakernet, 160, 172, 227-228, 361, 453
and social engineering, 172, 232, 252
as vulnerability points, 232
repeater based radio systems, 67, 379
repeaters, 67, 77, 378-379, 391, 393, 465
report-by-exception mechanism, 79
report-by-exception reporting, 451, 456
reports, as data-exchange mechanism, 127, 338
resistive temperature devices (RTDs), 25, 466
resource consumption malware, 188, 191, 276, 445
revocation, 207, 223, 303, 434
right-of-ways, 68, 391, 396
risk assessment
and mitigation, 211, 237-238, 244, 422
process of, 238, 244, 248, 411
risk self-assessment guidelines, 244, 248
rogue APs, 440
rollback capability, 302, 466
rootkit, 276-277
routers, 72-73, 79, 81, 85, 92-93, 184-186, 189, 200, 205, 222, 266, 292-293, 316, 395, 431, 447, 466, 472
RTU communications, 30, 51, 58, 63, 86, 220, 292, 374
RTU program logic, 394
run-time checks, 118
rural electric cooperatives (co-ops or RECs), 130, 370, 373

S

Sarbanes-Oxley (SOX) Act of 2002, 289
satellite communications, 70
SCADA systems, 191. *See also* alternative (backup) facilities
alternative operating site, 58, 364
application development for, 78, 101, 109, 115-118, 120-121, 212, 364
backup information generation and storage procedures, 303
communications connectivity to, 61, 72, 78-80, 172, 325, 342, 377, 389, 396, 438
communications security, 206, 225, 236, 292, 325, 343, 355, 358, 370, 374, 385-386, 423-424
cybersecurity issues in, 354, 358, 363, 366, 369-370, 374, 377-378, 380, 384-385, 393, 395-396
evolution of software, with commercial software, 101
functions of, 100, 109, 167
interoperability, 54
operator-console architecture, 99, 106, 144, 396
and plant DCS, data and control exchange between, 343, 364, 397

procedures and policies, 170, 213, 297, 304-306
production system modifications procedures, 301
remote access, 92-93, 238, 355
remote access to, 103, 172-173, 177, 179, 209, 388, 400
restoration procedures, 251, 302, 307-308
SCADA vendor use of, 171
secure socket layer (SSL), 91
securing remote users to, 213
security features, 54, 67, 89, 91, 104, 122, 346, 410
security patches for, 189, 243, 301, 305, 350, 426
start up and shut down procedures, 302, 429
system configuration of, 170, 212, 221, 271, 289, 350, 429
system physical security, 209, 240, 279-281, 289-291, 295, 356, 371, 396, 410
technical security strategies, 224, 349, 396
telecommunications interconnections in, 172
update process for, 305
user access to, 209, 353
vendors role in security of, 171, 221, 225, 231, 243, 276, 308, 349, 393
vulnerability elimination, 238, 240-241, 398-399
vulnerability points, 238
water/wastewater industries block diagram, 378
scalable systems, 14, 224, 270, 346
scanning and filtering, 189, 316
scrambled messages, 91, 214
scripting languages, 118-119, 121, 273, 275, 456
script kiddies, 255, 398, 467
second administrator, 208-209
second authorization procedure, 213, 289
secret passwords, 131, 208, 213, 445
secure protocols, 374, 385, 468
secure shell (ssh), 175, 186, 210, 358, 410
secure socket layer (SSL), 186, 207, 313, 328, 355, 372, 457
security issues, 94, 114, 280, 334, 386
security mechanisms, 65-67, 76, 102, 117, 125-127, 189, 204, 372, 439
security patches, 102, 189, 243, 301, 305, 350, 426
security perimeter, 240, 281, 289, 291-294, 311, 371, 416-417, 419, 435
security policies
development of, 298, 300
employee training in, 213, 306-307, 412-413, 439
enforcement of, 298-299
security related patches, 306
security risks, 35, 102, 203, 211, 335, 396
select-check-operate sequence, 9, 53, 385, 446
self-checking features, 316, 330
self-healing rings, 82-83, 366, 468
semi-automatically generated standard displays, 108, 136-137, 144
semigraphic operator display, 7, 211
sensitive material management, 303
separation of duties, 116, 384
sequence-of-events (SOE) recording, 26, 34, 46, 467
sequential-function charts (SFCs), 42, 50, 455, 467
serial (analog) communications
encapsulating and transporting over secure wide-area network (WAN), 238, 381
securing with encrypting modems, 378, 424
serial (RS232/C) connections, 75, 379, 384, 424, 470
serial line encrypting modems, 379, 424
serial ports, 4-5, 12, 31, 33, 59, 65, 74, 369, 382, 446, 451, 453, 459, 465
serial protocols, 5, 31, 33, 36, 50-58, 64, 174-175, 238, 373-374, 380, 382, 385, 445, 451-452, 470
servers, 14, 17, 65, 102, 169, 205, 224, 253, 266, 271-272, 275, 289, 302, 311, 315, 330, 332-334, 336-338, 342, 358, 364, 372, 395-396, 400-401, 404, 412, 435, 451, 457, 461, 473-474
Server Side injection, 272
service set identifier (SSID), 440, 469
session-hijack attack, 388
session-key message encryption, 91, 222, 455, 467

session keys, 424, 467
set points, 149, 390
sewage treatment plants, 50, 54, 153, 378, 382-383, 386-388
shadowing, 233, 306
shared secrets, 208, 213, 216
S-HTTP, 468
simple mail transfer protocol (SMTP), 319-320, 400
simple network management protocol (SNMP), 59, 133, 293, 333-336, 447
simplex channels, 64, 343, 468
Slammer (worm), 243, 400
Slave Remote Terminal Units (RTUs), 32-33, 447, 472
smart cards. *See* removable storage media
smart packet radio networks, 283-284
smart remote terminal units (RTUs)
  downloaded logic and parameters, 36-37, 284, 393
  vs. dumb remote terminal units (RTUs), 366
  evolution of, 36, 393
  in pipeline industry, 377, 389, 393
  technology of, 284, 393
sneakernet, 160, 227-228, 361
social engineering, 232
  actions observed through, 183, 232, 234-235, 252, 265, 279, 306, 345, 354
  described, 232, 468
  employees trained about, 235, 302
  incident response and reporting procedures, 233, 236, 304
  techniques used by attackers, 232-234, 434
software architecture, 20, 178
software layers, 100, 464
software libraries, 115, 117, 468
software updates, 116, 135, 173, 240-241, 290, 301, 429, 442
software versions of operating systems, 189
solar power, 42, 79, 389, 468
spam, 193, 276-277, 314, 428, 441, 469
special quality of service (QoS) requests, 96, 347, 357
spectral energy (frequency) distribution, 77, 380, 469
spreadsheet report, 166
spreadsheet report generators, 166-167
spread-spectrum radio, 76-78, 451, 453
sprinkler systems, 291
spurs, 392
SQL-compliant relational database packages, 123-124, 271-272, 469
SQL injection, 124, 271-272, 406
SQL injection attacks, 272-273
stack, 55, 57, 81, 126, 202, 249, 270-271, 316, 331, 457, 464, 469
Standard 1300—Cyber Security, 375, 415, 425
standard displays, 132, 136, 289, 455
standardized API's, 117, 121, 444
Standard Query Language (SQL), 123, 271, 469
standby generator, 294
start up processes, 148, 302
stateful inspection firewalls, 191, 316, 453, 469
static IP addresses, 182, 351, 469
statistical calculations, 166, 237, 300, 339
status inputs, 24, 26, 28, 33, 35-36, 38, 45, 52, 149, 158, 174, 199, 303
storage media, 160, 172, 303, 330
stream cipher, 216
strip-chart pen recorders, 156-157, 469
strong authentication mechanisms, 211, 303, 361, 372, 469
strong authentication technology, 131, 211, 361, 372
structured walk-through, 308
submaster, 32
sub-multiplexed voice-grade channels, 386
substation automation, 33, 47, 295, 369-370, 419, 470
substation data concentration, 33, 470
substation data concentrators (SDCs), 366, 369, 470
substations
  access to communications at, 366, 368, 371
  automation of, 369
  backdoors to, 366, 368-369
  configuration of, 368
  IP networking at, 365, 370-372
  VPN technologies at, 369-372
  vulnerability points of, 368
Sun/Solaris operating systems, 101
supervisory and local control applications, 383-384

supervisory control and data acquisition (SCADA) systems
architecture of, 12, 236, 248, 377, 390, 394, 411, 467
client-server architecture of, 14, 122, 390
component diagram of, 2, 225-226, 378
distributed architecture of, 12, 390
functions of, 109, 116, 132, 147, 157, 167, 333, 395
history of, 1, 363
information flow within, 21, 198, 344
software packages for, 111, 241, 334
technological convergence of, 15
supervisory control applications
operating system utilities, 102-103, 132
for pipelines, 394
program development tools, 212, 395
SCADA system utilities, 212, 305, 384, 388, 394-395
standardized API's, 444
supervisory control of local regulatory control panels, 39, 284
supervisory control script, 120, 384
supervisory control systems, 1, 39, 397
supervisory points, 37-38, 118
sweeping, 176, 463
symbols for measurement conditions, 154
synchronous optical networking (SONET), 69, 179, 386, 468
Synchronous Protocol, 4, 54
SYN flooding attack vs. firewalls, 190, 469
system administrators, 103-105, 208, 212-213, 411
system architectures trends, 178, 225, 395
system backup materials
access to, 212-213, 290, 303
off-site storage of, 290-291, 303
on-site storage of, 303
system backup procedures, 170
system configuration, 105, 121, 170, 212, 429
system event log, 165, 340
system host software layers, 313, 464
system-level programmer, 270
system operational status display, 134-135, 169
system physical security, 209, 240, 279-280, 290-291, 295, 324, 326, 356
system programs and supervisory applications, 377, 389, 394-395

T

T1 circuits, 62, 69, 72, 371, 437-438, 448, 470
T3 circuits, 70, 72, 292, 371, 437-438, 470
tabular operator display, 7
tag bypassing, 155, 219, 307
tagging (safety), 155, 201, 395, 463
tagging display, 156, 211
tag names, 109, 113, 119, 470
tag/point database building, 110, 463
tags (XML), 474
targeted attacks, 232
TASE.2, 54, 126, 372, 455
TASE.2/ICCP connections, 372-373
TCP (transport control protocol), 14, 17, 55-59, 63, 65, 73, 84, 86-87, 90-91, 93, 95, 97, 123, 126, 135, 178, 183, 185, 188-191, 196-197, 249, 268-269, 275, 315-316, 320, 329, 331, 343, 351-352, 355, 409, 448, 469-470
TCP finger printers, 249, 470
TCP/IP (transmission control protocol/Internet protocol), 177-178, 470
architecture of, 178, 448
backbone/MBONE, 185
concurrent communications, 196, 296, 324
described, 178, 470
encapsulation, 452
vs. frame relay, 181
functions of, 296, 470
ICCP protocols, 365, 372-374
Internet service providers (ISPs), 457
IP address sweeping, 176, 463
IP spoofing, 176, 186, 443, 457
IP suite (of protocols), 177-179, 189, 449
IPv4 (IP version 4) and IPv6 (IP version 6), 176-177, 179-180, 182, 184-186, 188, 195-196, 359, 457
vs. ISO/OSI model, 372, 464
networking of Microsoft Windows, 17, 97, 182
remote access, 182, 209, 388, 395
secure socket layer (SSL), 313, 355, 372
VPN, 365, 372, 382
vulnerabilities of, 189, 249, 269, 275, 357
TCP/IP (transmission control protocol/Internet protocol) Suite, 177, 189
TCP messages, 249

technical security strategies
configuration data, 335, 349
electronic perimeter, 311, 416
intersystem and intrasystem data exchanges, 122, 177
operational access, 388
RTU communications, 220, 374
system programs and supervisory applications, 377, 383, 389, 394-395
telcom room, 396
telecommunications, 1, 3, 42, 212, 291-292, 303, 371, 419
telecommunications interconnections in SCADA systems, 212, 291-292
telecommunications technologies
commercial voice/data carriers, 202, 296, 380, 386, 391, 457, 461
digital networking technologies, 234, 371, 378-379, 385-386, 424, 457
Internet, 298-299, 303, 370, 457
TCP/IP networking, 176, 285, 296, 470
voice-grade (analog) telephony, 391, 459, 473
wireless communications options, 194, 206, 472
telephone circuits to bridge local area networks (LANs), 73, 383
telephone company algorithm, 44
telephone technology, 3, 61-62, 65, 68-69, 71, 75, 78-79, 81, 83-84, 172
telnet, 103, 210, 358, 410
temporary digital certificates, 222, 303
test and development system procedures, 308
thermoelectric generation, 471
thermoelectric generators, 471
thin client, 388
threat assessment and countermeasure selection, 189, 248, 311, 422
threats and threat agents
defined, 230-231, 468
external threats, 230-231
insiders as, 230
internal threats, 230
and most probable objectives from an attack, 229, 251, 311
targeted attacks, 232
threat scenarios and potential outcomes, 149, 171, 279-280, 393, 411
threat sources taxonomy, 229
time bombs, 116, 384
time-stamp devices, 26, 46, 126, 283
time synchronization, 44-45, 341-342, 348, 365, 385
time to live (TTL) counter, 189, 471
tokens, 132, 210, 234, 342, 434
token schemes, 132, 210
top-down and bottom-up configurations, 47
totalizers, 28, 471
training
of customers, 336
for employees, 213, 302, 306-307, 412-413, 439
of employees, 212, 231, 290, 298, 301, 306, 308, 411, 416
transceivers, 32, 444
transducer-less AC inputs, 46-47
transducers, 46, 285, 288, 295, 326, 345
transmission control protocol (TCP), 14, 470
transmission control protocol/Internet protocol (TCP/IP). *See* TCP/IP (transmission control protocol/Internet protocol)
transmitter, 3-4, 66-67, 292, 379, 453, 471
transport-layer security (TLS), 207, 243
transport services, 55, 232, 377
trend displays, 161, 211, 334, 461
trend groups, 158
trip-close pairs, 27, 365
trivial file transfer protocol (TFTP), 194, 209, 269, 293, 454
Trojan function, 116, 276, 454
Trojans, 276
trunked radio systems, 67
trusted platform module (TPM), 220
two man rule, 319

U

UCA1.0, 54, 126, 373, 455
UCA2.0 protocols, 34, 54, 58, 80, 127, 373, 381, 471
UDP (user datagram protocol), 57, 87, 90, 185, 205, 249, 316, 320, 331, 351-352, 409, 448, 472
UDP messages, 87, 189, 316, 343, 355
UHF/VHF radio, 65-66, 379, 472
unacknowledged alarms, 150

uniform resource locator (URL), 89, 193, 314, 358, 451, 457, 462, 472
uninterruptible power supply (UPS), 170, 289, 294, 334
universal asynchronous transmitter (UART) chip, 4, 445-446, 471
Unix/Linux operating system, 59, 358, 361, 472
unsolicited report by exception communication mode, 374, 451, 456
unsolicited report by exception scheme, 374, 451
Urgent Action Standard 1200, 375
user, 10, 14, 19-20, 25, 34, 37-38, 41, 74, 76, 93, 95-96, 130, 167, 169, 172, 189, 193, 204, 208-209, 211-213, 218-219, 222, 226, 233, 243, 253, 265-266, 268, 272, 274, 276-277, 285, 314, 319, 322, 327, 333, 336, 338-339, 349, 351-353, 357, 364, 377, 379, 410-411, 413, 421, 436, 443, 445-447, 452, 454, 456-458, 462-463, 466-467, 472-473, 475
user access, 1, 103, 105, 108, 191, 207, 209-210, 351, 353
user account management utilities, 106-107, 132, 208
utility communications architecture (UCA), 34, 373, 471
utility information bus (UIB), 168, 363
utility programs, 38, 47-48, 100-101, 104, 115, 132, 219, 446, 462

V

VAX/VMS operating systems, 101, 253, 311, 412, 473
VBscript, 104, 118-119, 166, 272, 467
vertical integration, 390
very-frequency (VHF) radio transmissions, 65-66, 76-77, 379, 472
virtual architecture, 178
virtual circuits, 74-75, 83, 472
virtual LAN (VLAN), 196, 201-202, 204, 225, 285, 387, 472
virtual local area networks (VLANs), 195, 200, 204, 293, 357
virtual remote terminal units (RTUs), 32, 472
viruses, xxi, 76, 86, 255, 276, 314, 449, 472
virus-scanning management, 301
virus-scanning software, 301, 435
Visual Basic (VB) scripts, 139, 273
voice-grade (analog) telephony, 55-56, 61, 75, 391, 473
voice-grade channel, 62, 67-68, 386
voice-grade telephone communications, 62, 68, 75, 386, 391, 473
voice over IP (VoIP), 62, 71, 78, 96, 216, 267, 391, 395, 460, 473
VoIP telephone service, 62, 267, 391
VPN configuration for remote access, 92-93, 131, 226, 410, 473
VPN technologies
  authentication/validation process required by, 92-93
  benefits of, 93, 173, 210, 355
  for communications security, 93, 355, 372
  described, 92, 355, 473
  and firewalls, 94, 186, 313, 315, 320, 356
  limitations of, 296
  occasional connection vs. permanent architecture, 93, 401
  reliability of, 355
  for remote user authentication, 93, 131, 210, 223, 417
  required by, 355
  at substations, 370-371
  with temporary digital certificates, 210, 223
  threats not protected by, 355
  and VPN tunnels, 93, 382, 388, 473
VPN tunnel
  mechanisms for establishing, 93
  for remote user authentication, 473
  security provided by, 382, 388
  threats not protected by, 93
  vs. VPN client/server design, 93
vulnerability elimination for SCADA systems, 241, 256, 399, 410
vulnerability points
  Bluetooth-enabled devices as, 194
  of commercial hardware, 241, 349, 372
  dial-in telephone lines as, 173
  of distributed common object model (DCOM), 349
  of firewalls, 249
  to insider attacks, 203, 230-231

of intelligent electronic devices (IEDs), 282, 284, 288
local loops as, 359
of Microsoft Windows operating system, 210, 403
of OPC, 313
of operating systems, 210, 349
of relational databases, 272
removable storage media as, 298
SCADA historian/trending packages, 248
SCADA systems, 189, 231, 398-399
of SCADA systems, elimination of, 238, 240-241, 256
of substations, 295
of TCP/IP, 249
threat and threat agents, 230-231, 311
of Web site/server, 275
WiFi APs as, 174
WiFi equipped devices as, 174

W

wallpaper, 139, 141, 473
war dialers, 173-174, 368
war driving, 174, 232, 473
warm-backup schemes, 12
wastewater, 237, 364-365, 377-382, 384-388, 390, 392-395
watchdog timer, 11, 474
water treatment plants, 387-388
water/wastewater industries
licensed radio communications, 65
municipal LANs and wide-area networks (WANs), 79
programmable logic controller (PLC) equipment as remote terminal units (RTUs), 40, 49-50
remote terminal unit (RTU) requirements of, 36, 38-39
Web-based HMI architecture, 271, 400
Web-based HMI servers, 271
Web browsing, 92, 191, 209, 218, 227, 251, 273-274, 298, 312, 332, 351, 357, 360, 378, 388, 396
Web browsing and e-mail integration, 191, 209, 218, 273, 312, 332, 396
Webcam technologies, 395, 474
Web pages and logical relationship between relational databases, 169, 272, 469
Web radio, 90, 96
Web site/server
denial of service (DoS) attacks against, 191, 276, 403
of hackers, 101, 173, 183, 253, 255, 268, 454
list of forbidden, 193, 313, 427, 445
malware sent from, 186, 193, 276, 331
SCADA vendor use of, 231, 243
secure socket layer (SSL), 91, 355
SQL commands from, 243, 272, 469
SQL injection vulnerability of, 272-273, 469
vulnerabilities of, 189, 234, 266, 268, 272
white-hat hackers, 172, 452
wide-area networks (WANs)
encapsulating and transporting over secure wide-area network (WAN), 316
fiber optic network wide-area networks (WANs) and UCA2, 65, 374, 386
IP wide-area network (WAN), 15, 76, 84, 177, 371, 374, 381, 395
municipal LANs and wide-area networks (WANs), 378, 381, 383, 386-387, 459
SCADA systems and, 172, 177, 305, 316, 374, 386
serial (analog) communications on, 238, 381
WiFi, 35, 76, 92, 174, 184, 192, 194, 207-208, 216, 298, 381, 428, 435-436, 440-441, 444, 452, 455, 458, 466, 469, 473-474
WiFi ad hoc networks, 192
WiFi APs
protection of, 206-207
as vulnerability points, 174, 207, 466
WiFi equipped devices, 458
WiFi protocol access (WPA and WPA-2), 174, 207
WiMAX, 78, 378, 474
wired and wireless digital networking, 78, 371, 378, 385
wired equivalent privacy (WEP), 207, 266, 436, 452, 474
wireless access points (APs), 174, 204, 207, 397
wireless communications options, 76, 78, 194

wireless Ethernet, 76, 78, 204, 206, 208, 293, 444, 474
wireless fidelity, 35, 76, 92, 174, 184, 192, 194, 206-208, 216, 298, 350, 428, 435-436, 440-441, 444, 452, 455, 458, 466, 469, 473-474
wireless local area networks (LANs) (WLAN)
    administration and supervision of, 207
    defense of, 206-207
    physical security of, 207
    prevalence of, 206
    SCADA system separation from, 206
    SQL injection against, 272
wireless networking capability, 78, 194, 378, 380, 385
Wonderware (commercial software), 14, 122
worms
    in attack messages, 255
    and Bluetooth-enabled devices, 194
    DCOM and, 255
    defense against, 252
    infection rate of, 276
    as malware, xxi, 251-252, 276, 401, 474
    prevalence of, 86, 277-278, 401
WWV broadcasts, 45

X

X.25 packet switchers, 70, 74, 437, 474
X.509 format, 222-223, 444-445, 450, 474
X11 protocol, 474-475
X-terminal, 210, 474-475
X-terminal devices, 210, 475
X-Window(s)
    for centralized operator-console architecture, 474-475

Y

yagi antennas, 380

Z

zip disks. *See* removable storage media
zombie computers, 191, 451, 475
zooming, 142, 145, 147
zoom in graphics, 142-143, 145