There is a serious difference between a developer and a hacker. The developer focuses on the faultless and performance-based operations of the system or software. The hacker, on the other hand, focuses on the vulnerabilities in that system and how she can exploit them to her benefit. The fundamental difference is one of perception.

When we examine the API features of the web application we test, we see that it has the feature of adding logs manually. We understand that it is possible to produce tens of thousands of fake and misleading logs.

We add this vulnerability to our report along with the screenshots. Wolves love hazy air, and log contamination is a method of making the air hazy.

BILISHIM

Cyber Security & Artificial Intelligence Sevices

Yilmaz DEGIRMENCI

PENTEST DIARIES

PENTEST DIARIES

BILISHIM

Cyber Security & Artificial Intelligence Sevices

A PENTESTER'S DIARY

YILMAZ DEGIRMENCI

Bilishim

Cyber Security & AI

A Pentester's Diary

Foreword

Penetration testing involves examining information systems from the perspective of a hacker and exploiting the discovered vulnerabilities, which are then documented. Once the report is published, the systems are tested once again to ensure that the mitigating steps were properly performed. An expert in this area is Bilishim Cyber Security, a TSE-A and EMRA-certified penetration testing firm. Every test in this highly sophisticated study has its own story. This journal is made up of real experiences we have had during the course of our careers. Each test is full of enthusiasm because we enjoy our work so much that we are delighted to share it. I would also like to thank my distinguished coworkers, with whom we shared these experiences.

Das Tagebuch eines Pentesters

Vorwort

Bei einem Penetrationstest werden Informationssysteme aus der Perspektive eines Hackers untersucht und die entdeckten Schwachstellen exploitiert, die anschließend dokumentiert werden. Nach der Veröffentlichung des Berichts werden die Systeme erneut getestet, um sicherzustellen, dass die Maßnahmen zur Schadensminderung ordnungsgemäß durchgeführt wurden. Ein Experte auf diesem Gebiet ist Bilishim Cyber Security, ein TSE-A- und EMRA-zertifiziertes Unternehmen für Penetrationstests. Jeder Test in dieser Arbeit hat seine eigene Geschichte. Dieses Tagebuch besteht aus realen Erfahrungen, die wir im Laufe unserer Karriere gemacht haben. Jeder Test ist voller Begeisterung, denn unsere Arbeit macht uns so viel Spaß, dass wir sie gerne teilen. Ich möchte mich auch bei meinen verdienstvollen Mitarbeitern bedanken, mit denen wir diese Erfahrungen geteilt haben.

Дневник Пентестера

Предисловие

Пентестирование включает в себя изучение информационных систем с точки зрения хакера и использование обнаруженных уязвимостей, которые затем документируются. После публикации отчета системы еще раз тестируются, чтобы убедиться в том, что меры по смягчению последствий были выполнены должным образом. Экспертом по пентестированию является Bilishim Cyber Security, компания по тестированию на проникновение, сертифицированная TSE-A и EMRA. Каждый тест в этом очень сложном исследовании имеет свою собственную историю. Этот дневник составлен из реального опыта, который мы получили за время нашей карьеры. Каждый тест полон энтузиазма, потому что нам так нравится наша работа, что мы рады поделиться. Я также хотел бы поблагодарить моих уважаемых коллег, с которыми мы поделились этим опытом.

WHAT IS PENETRATION TESTING?

A penetration test, also known as a "Pentest" for short, is essentially a highly specialized testi ng process that necessitates advanced engineering skills in order to scan, analyze, infiltrate, an d harden against cyber attacks in all of the components of an information system.

During the testing phase, experts identify all security vulnerabilities and risky system element s from the perspective of an attacker.

Pentests should be performed by experts due to the variety of methods and factors.

WHY IS THIS IMPORTANT?

The ultimate goal of a pentest is to understand how secure your system infrastructure is from an attacker's point of view and to remediate any vulnerabilities encountered. In addition to identifying an organization's weaknesses, it is critical to assess the relevance of its security policy, evaluate staff awareness of security issues, and determine the extent to which cybersecurity principles have been implemented. The most important thing to keep in mind is that all systems are vulnerable. No system has ever been entirely secure. In addition to these primary motivations, it is critical for organizations to have their systems pen-tested on a regular basis and to have their operating systems and applications updated by using the necessary security patches at regular intervals.

Penetration tests are divided into three stages:

Black Box: No additional information is requested from the institution where the tests are conducted. Using manual methods the system is attempted to be hacked from the institution's external interfaces. External threat vectors can thus be detected. Black box testing is the closest test method to a real-world attack situation. The primary purpose is to gain access to the target system's database.

Gray Box: By visiting the institution as a guest, the attack surface of the system is investigated by exploiting loopholes. Many flaws are discovered during the gray box testing phase, such as the attacker's access to the company's IT infrastructure and its databases. The Gray Box approach illustrates how successful an attack by a hacker with physical access to the organization can be. Even with limited authority, the ultimate goal is to take over the entire system.

White Box: In the organization where this test is being run, the security expert wants an account with the privileges of an ordinary user. As a result, the expert, as a standard user, attempts to discover and take advantage of system flaws, allowing him to escalate his privileges or access services that he is not normally authorized to. Various tools, as well as particular manual procedures, are used during this testing phase, including the essential strategies that prevent the system from identifying the attacker. The White Box approach demonstrates the effectiveness of an internal attack on a company.

THE STAGES OF PENETRATION TESTING

Penetration testing is a process that follows a specific systematic methodology.

1. Passive Reconnaissance and Fingerprinting

The scope of the test and its objectives, as well as the systems to be examined and the test techniques that need to be applied are determined. This stage gives information about all kinds of details that may result in a successful attack, such as domain names, network blocks, routers, IP addresses and employee private information, and phone numbers within the target system infrastructure and scope. In some circumstances, information obtained through open sources might be extremely vital. Numerous assets are utilized for this purpose, particularly the target institution's website and social media.

2. Active Reconnaissance and Scanning: During this phase, it is feasible to make a deduction by leveraging information gathered in the previous stage, such as identifying the active devices in that particular IP-Block. It is crucial to determine the operating system installed on the discovered live devices, the open ports, and the running services, as well as their version information. Furthermore, the network traffic can be monitored in order to gather as much important information about the system infrastructure as possible.

3. Vulnerability Assessment

After numerous intrusion attempts on the target system's apps and analyzing the responses returned, connection requests will be made to the active services.
In simple terms, it is the stage at which services like ftp, netcat, and telnet are used to interact with the intended applications.

4. Exploitation

All of the information gathered in the prior steps serves a single purpose: gaining unauthorized access to the target system, extracting records from its database, or acquiring information that should otherwise be inaccessible.

The target's operating system, open ports, services running on these ports, and exploit methods that can be used based on their version are all investigated in this step, hence the system is attempted to be infiltrated. Because of the existence of additional attack vectors, external web-based portals and apps are more vulnerable. As a result, they are more prone to attacks.

Taking over a system necessitates extensive knowledge and experience, as well as the ability to use existing exploitation methods successfully and flexibly without causing damage to the system or leaving any traces. As a result, the exploitation phase is the most crucial and critical phase of a pentest.

5. Privilege Escalation

The strength of a system is determined by its weakest link. In the first instance, access to the system is typically granted at a low degree of privilege. At this point, the PenTest specialist aims to gain administrator privileges by exploiting vulnerabilities in the operating system or environment, then to use these additional privileges to gain access to other network devices, and finally to the highest level of user privileges, such as domain administrator or database administrator.

PENETRATION TEST SERVICE AND AFTERWARDS

As a result of the penetration test, the vulnerabilities discovered by our company in the target systems and how these vulnerabilities were exploited are documented in the "Penetration Test Result Report", which is presented with sample screenshots. The final report also includes a mitigation recommendation for each vulnerability.

To preserve the sensitivity, independence, and integrity of the penetration testing we undertake, we do not specifically sell products as a cyber security company. However, if desired, we offer secure network architecture design and hardening consulting services to increase the trustworthiness of information systems.

PENTEST DIARIES

Seeing the Unseen

The days when I first went into the field in terms of pentesting, we were assigned to a test at a software company in Teknokent. As always, my first step is to monitör the network traffic. I was told that there are 3 IP blocks starting with "192.168", as shared by an IT admin friend. He also stated that there is a private 10-block IP block, that only he has access to, and that we do not need to scan it as part of the test.

When the Eternalblue vulnerability was still fresh. I discovered the vulnerability on one machine and captured an authorized username and password over RAM. Then I try it on other servers using the pass-the-pass method and take over dozens of servers.
The IT manager friend who was watching the situation with me is quite confused.
I also collect data by finding a Postgresql database left with default permissions (which I've come across many times in many places).

But the most fun step comes after that. I found out the IP and MAC address of your IT admin friend. I use them to change the IP and MAC of my own computer. I see that I can also reach the 10 IP blocks that it could not reach before, and I identify the vulnerabilities there and add them to the report.

2 years later, while chatting with different friends from a completely different industry (one of them was a mutual friend of both me and the IT manager), the subject came to this pentest. We understood that a friend of the IT manager mentioned to many people about that pentest.

Hole in the Wall

A situation we encounter with many of our clients. There's an original software that dates back a long time, and it's running on an old Windows 7 or Windows 2008 server. Because the organization's staff is afraid of breaking the software, they never touch that machine, and a potentially deadly vulnerable machine like Eternalblue continues to exist on that network.

We're starting our test. Out of about a hundred and thirty machines, only one has this vulnerability. Our client is actually aware of the vulnerability; it was discovered in a pentest conducted by another company last year. However, they haven't considered it much of a concern because it's just a single machine.

After taking control of the machine, we seize a special account like "backupadmin" through the RAM area. Then, within an hour and a half, we take over 69 machines. We connect to most of them via Remote Desktop and capture screenshots. Our client now understands the seriousness of this vulnerability much better. We learn that this machine, which has been running for years, was shut down and removed from the system within an hour.

Finding the Needle in the Haystack

Those were the first days when I started to provide pentest service. Our client is a local but internationally operating software company focused on product development. They had previously conducted a pentest, but they weren't satisfied. They wanted to do it again.

I have daytime training sessions going on, so I'm working until late at night. The application has a vast surface, with dozens of form pages on the portal. I meticulously examine every corner for two weeks. Finally, the profile page of an application catches my attention. While changing my profile picture, I send a .txt file. By reading the PATH value, I detect where it goes, and I indeed find that I can access the content of that file.

It's nothing other than a classic "file upload" vulnerability. It's a PHP page like DVWA. My teammate attempts to deploy a ready-made webshell, but it doesn't work. I quickly prepare a "one-liner" shell and upload it. Bingo, I get a shell. Then, I start to discover internal network by running the "arp -a" command.

This vulnerability is somewhat easy for a pentester, but the real challenge is to detect the presence of such a vulnerability in such a vast environment.

Seeing that a remote shell can be obtained greatly impresses the software company. As for me, I'm experiencing firsthand how crucial it is for Secure Software Development to be understood by developers themselves.

Nettarsier

We are conducting a penetration test for one of Turkey's most important public institutions. The institution's network is so vast that our scans are taking months to complete.

During this process, we are also utilizing a well-known vulnerability scanning tool (from the Pro version). It's proven to be very helpful, but I feel that we're hitting some roadblocks in certain areas.

Due to factors such as the immense size of the institution's network and the recent transfer of some authorized personnel, I realize that even having control over the inventory is a challenge. I observe the need for mapping – knowing which components are in which network block, which services are active, etc.

I think by myself, "What if, while scanning the network block, a clean dashboard could present all services individually as separate packages on a single screen? For instance, FTPs and anonymous accesses, separate databases, web portals, NFS, webdav, active directory machines, and so on."

I gather opinions from various penetration testing experts. They all express that this idea is quite logical and would be highly beneficial.

After establishing the Bilishim company and opening our office, we decide to project this observation with our small team, and we start working on it. This leads to the creation of our Corporate Vulnerability Scanning Tool named "Nettarsier." With features like periodic scanning and task assignment, Nettarsier transforms into a highly functional product that can be used in Cyber Operation Centers.

By the way, a tarsier – known as "maki" in Turkish – is the largest-eyed and cutest forest animal in the world, with eyes proportionally larger than its body.

Exploration of Cache

A foundation of trust-based friendship often forms between us and the clients we conduct penetration tests for. A client who was satisfied with our pentest from a year ago expressed their desire to work with us again this year. We completed the necessary processes and began the pentest. Our client, whose solutions are Java-based, especially due to the Log4j vulnerability, requested us to carefully examine Java SpringBoot vulnerabilities.

As we conducted our scans in this manner, we indeed identified two critical SpringBoot vulnerabilities and reported them. One of these vulnerabilities was related to accessing heapdump files.

Our client, who is also a skilled programmer, became interested in this finding and examined the heapdump file. Here, they discovered that they could read the JWT token secret key value in plaintext. This finding also captured my interest, and together we continued to examine the heapdump. After a while, when conducting searches with various specific wildcard values, we realize that we could read usernames and passwords in plaintext as well. Together, we discovered the extent of the critical nature of this vulnerability. We genuinely witnessed firsthand how environments like cache, environment variables, and heap dumps should ideally remain as closed off as possible from the external world.

Once again, I witness how crucial it is to know about software architecture in the context of cybersecurity.

Who Said That We Can't Add Our Own Parameter?

I'm testing a web portal that has undergone regular pentests and is well protected by a WAF. In fact, every manipulation I try during any request process seems to be ineffective. I spent hours continuously trying various methods, but I couldn't find anything significant.

I continue to explore the features of the portal. There's a "Invite a Friend" feature on one tab. First, I send a functional real test message to my own email address. Along with a message content, I receive a referral link. Then, when I resend it, I analyze the traffic going through a proxy. Unfortunately, there are only email and message parameters, and I can't manipulate the outbound link.

I wonder what would happen if I added a "link" parameter and decide to test it this way by sending it through a repeater. In the received email message, I notice that the link address that should normally be there is missing. Therefore, only the content of the message I sent is visible. This leads me to realize that a fake link I could include in this message could be successfully used in phishing attacks.

I observe that the most interesting findings are often identified through small scenarios created by understanding the workflow.

Understanding the System is the First Step to Seizing It

We're finally back at the office after performing an ICS pentest out of the city. After our team members went to their homes, I began to examine another web pentest. I focus on profile picture uploading for a while. I have been trying methods like Exif, but I was not getting any results. Then, I notice that this site automatically generates a PDF membership certificate for its members. I know that PDF generation scripts often have serious vulnerabilities, so I take a look at the source code. I realize an interesting URL there. I understand that there's a link generating a QR code beneath the membership certificate. When I visit the link, although it seems complex at first, after cleaning it up a bit and decoding the Base64 part, I understand that it's actually an address that calls a QR code based on the membership number. Naturally, I try encoding a random value in the same format using Base64, and indeed, a QR code corresponding to that number is generated. It seems like a way to create fake documents has opened up.

Hours pass, and I realize that I haven't even taken off my jacket yet.

Surveillance of Contestants

Our client contacts me, stating that despite all the precautions they've taken in their mobile application, they suspect that the race condition vulnerability has been exploited.

We examine the situation together, and after conducting necessary simulations, we don't observe any negative outcomes. We delve into the code together. If a user is buying a product worth 5 TL and a normal user might buy 20 of them by paying a total of 100 TL, if a single user is making 20 consecutive purchases, there's likely an incorrect pattern there. We emphasize the need for a mechanism within the code that detects and, if necessary, blocks such behavior. In a situation like this, we suggest implementing a mechanism that generates an error code like "TOO_MANY_REQUESTS" and simultaneously tracks user-based request attempts with a parameter like "nextAttemptinSeconds."

Never Give Up

The most sensitive panel of any web portal is naturally the account panel, where account and identity information is managed. Any slight vulnerability you detect here may directly cause to an account takeover or privilege escalation attack.

With this in mind, I'm examining the account profile page via the API through a proxy. When I try to update it, I'm only allowed to update the email address. Still, even with the least privileged user session, I proceed to send a POST request by adding parameters such as the username, last name, password, and role that I obtained from other user management traffic. However, my POST request is not accepted. I send requests targeting other users, but they are rejected as well.

This time, I update the POST request to a PUT request and try again. Users can't normally change their name and last name using POST and would receive a 400 error, but they can change it using PUT. This small progress excites me. I immediately attempt to grant admin privileges as a role, but this method fails. I try to modify data for other users, but it doesn't work. I think of creating a new account with an existing email address, but it doesn't allow me.

I take a short break to think about what else I can do.

Dear Cinematographer, Please Replay It Slowly

The penetration test we're conducting has a vast surface area with many sub-portals. While examining this surface, our teammate discovers a portal named "filemanager." Normally, the portal interface comes with a login screen, and it seems impossible to access without knowing the username and password.

Our teammate decides to dig deeper. When they individually intercept the traffic, they realize that they can actually use functions located in the top right corner without needing authorization. First, they try uploading a file using the upload function. The server returns a "server error" message. Subsequently, they attempt to create a folder using the createFolder function, and once again, they receive a "server error" message.

Although they haven't achieved the desired result in the initial stage, they continue to explore other ways to exploit this "path disclosure" vulnerability they found.

Money Goes Where You Transfer It

Today, I read that a vulnerability on the Coinbase website earned a reward of $250,000. It triggered memories of some of the pentests we conducted. It was during the early days of my journey in the field. A young friend who had made a substantial amount of money from Bitcoin trading had launched a Bitcoin trading platform. They reached out to me for a penetration test.

Firstly, when I ping the site, I saw that the IP address was not coming, I guess they have imposed some kind of restriction. I easily found out the IP address with a simple wget command. When I forwarded the IP address to my friend, he was quite surprised and wondered how I found it. Frankly, I didn't know how to explain this.

When I examined the PHP-based site, I felt that there was a weakness especially in SQL queries. When I put the traffic I captured with Burpsuite into the sqlmap query, I realized that I had actually captured the database. It was an interesting feeling to see something I always do in a laboratory environment in real life.

A couple of years later, I remembered that friend and called him to ask how he was doing. He sold the site to someone else. However, the saddest part was that his partner stole the wallet worth 500 thousand dollars and fled to Bulgaria. He returned to his main profession, optician. But I sensed from his tone that he was more at peace now.

Thanks to the software developer friend who developed that site (who is still a close friend of mine), whom he also did other IT works and brought their pentests to me. One of these projects was both a trading platform and an IOC project. One of the most interesting vulnerabilities here was that entering a negative value during a money transfer would actually increase your balance instead of decreasing it. Another vulnerability was that a person was not prevented from sending money to himself. A serious vulnerability in payment systems could lead to the entire system collapsing in a single day.

Next In The Line Please!

In Turkey, we frequently witness data being pulled, dumped, and sold on the dark web. Often, it is assumed that this is done through SQL Injection to dump the database. However, I believe, the most probable method of bulk data dumping involves exploiting Insecure Direct Object References (IDOR) and Rate Limiting vulnerabilities together, allowing sequential extraction of data nowadays.

Today, I'm conducting a pentest on a significant portal. Naturally, my initial focus is on user and identity management matters. I'm examining the details of a corporate user via a proxy. I notice that critical data, which is masked on the portal, is easily readable through the proxy. What's even more important, I realize that this corporate user data is being fetched numerically.

Then I perform this operation numerically and sequentially. It's not much of a surprise to see thousands of users' data flowing in front of me, one after another. The principle that the devil is in the details comes to my mind. I smile while taking the screenshots.

Sometimes You Have To Use Side Roads On The Main Road

Speaking of "the devil is in the details," attacks always exploit the weak spots of a system or application. I'm conducting a test on a new HR web portal. The login and identity information are quite well protected, leaving me with limited options to move forward. While navigating within the application, I want to see my own payroll information. At this moment, a pop-up screen requests my username and password again. My eyes light up because there's no CAPTCHA protection in sight.

I quickly enter the username and password and redirect the traffic through a proxy. Then, I perform a dictionary attack on another test user. Bingo! When it matches with the correct password, the packet size changes. I can potentially obtain someone else's identity information.

What if My Ticket Is Valid?

The success of a penetration test is directly related to understanding the workflow and logic of the system. That's why sometimes it's beneficial to step back, lean back, and switch to a bird's eye view. While examining a web portal, I notice that the user already has access to all permissions and entities. I request my client to create and share a less privileged user with me.

Then, I try out the following scenario:

1. Log in with the less privileged user and store the cookies in the proxy (for keeping session information).

2. Log in with a user who has administrator privileges, navigate to the address containing critical data, and also capture this traffic in the proxy.

3. Paste the cookies of the less privileged user into this captured traffic and resend it. I observe that I can access the same data.

As a result, by showing that a less privileged user can achieve results if they know the target address, I demonstrate the potential of the approach.

To Whom Will the Letter Go?

One of the vulnerability surfaces we often encounter in web portal pentests is the messaging feature. Developers often focus on functionality and may not take into account the possibility of an attacker sending fake messages.

During a test on a portal, I intercept the "Contact Us" option through a proxy. In the message I send, I manipulate the fields for Name, Email, and Phone Number, which are normally read-only and unchangeable. I can change these as I wish and easily send a fake message to the other party.

Especially vulnerabilities like this, which could involve insults or legal implications, should always be considered carefully.

Self-Organizing Databases

Today, we're conducting a pentest on a relatively large network. The systems are new, and the responsible colleagues are quite attentive, so we couldn’t find very obvious vulnerabilities. Nevertheless, we proceed to run nmap queries that scan the relevant ports of database systems within the network blocks.

As we expected, gaining access to MSSQL (and also Oracle and MySQL) databases with default credentials isn't possible. However, we find that we can access Mongodb, which records GET and POST requests, and Redis, which acts as a database cache, through their respective ports. We encounter a similar situation in many places. Sometimes, we're able to include PostgreSQL in this category as well.

We capture screenshots for our report, intending to share our findings.

Visa Required Pass

I am developing the software pentest for a corporate web portal. There is an ability to log in using official authorization via oAuth. In this situation, I am considering what I can do, and three different scenarios come to my mind:

-Replicating all steps of authorized account login and oAuth processes, including manipulating redirectUrl and parameters.

-Replicating with an unauthorized account, attempting to mimic authorized login traffic.

-Attempting to log in with a fake password from an account known to be authorized.

I rolled up my sleeves to try all these scenarios and continue the tests from where I left off.

Stealing An Identity Card

Holiday is a time of rest for most job sectors, a time to perform tests for some of our special and sensitive clients, hence a working time for us as well. Therefore, apart from routine visits, I constantly perform pentests.

A guest who also actively conducts pentests is visiting me. Out of his curiosity, he asks that "Don't you find yourself repeating every time, or don't you get bored?" I reply, "No," because each pentest surface holds a distinct flow and a separate story. Searching for vulnerabilities by reading that story is a different experience every time."

I begin to examine the password reset link of the web portal. When I click the link, I see that only the USER_ID value comes at the end of the portal address. Moreover, it doesn't expect me to know the current password directly. However, the USER_ID value isn't a value that can be guessed or found by trying sequentially. I continue my research.

I browse with an admin account and record and review the important traffic. My goal is to identify important API addresses. After quite a few attempts, I finally capture a GET request traffic that also provides the USER_ID value of users. Then, I log in with the unauthorized user and send the same GET request with its Authorization Bearer value. Since there is no authorization check, I can see the USER_ID value of all accounts, including Admin.

With this information, I can perform account takeover to any account, including Admin, by renewing the password.

I Have An Entry Certificate

I firmly believe that software developers should definitely receive "Secure Software Development" training. This training should be entirely hands-on, and especially at a basic level, it should demonstrate how web vulnerabilities are exploited by a hacker, even if only on a fundamental level.

In recent days, during my pentests, I've noticed that particularly in file uploads, only superficial extension checks are conducted, and this can be easily bypassed. In such cases, an uploaded web shell often becomes ineffective due to the presence of the EDR solution on the server.

Exploiting file vulnerabilities actually consists of two layers. First, being able to send the malicious file to the target system, and second, locating and accessing that file. Designing software architecture with these considerations in mind is crucial in this regard.

Even though I realized that the ASP shell I deployed today was deleted, the fact that the location information was clearly present in the response text indicated that I needed to dig deeper into the matter. To take a concrete step, I first uploaded an HTML file containing an XSS payload and easily executed the XSS attack.

While contemplating what alternative I could use instead of ASP, I decided to utilize a web shell with a .cer extension. After tweaking it a bit to alter its signature, I sent it. Bingo, I could navigate through the server's directories.

Simple and effective methods.

Great Power Requires Great Responsibility

Today, we received the files of their corporate mobile application from our client, and we begin testing. The application identifies individuals using their National Identification Numbers (NIN) and facilitates assigning tasks to them. While our team member was examining the application's functions, he changed the NIN using Burp Suite and provided his own father's NIN. To his surprise, he found that the real information associated with that NIN was retrieved. Consequently, he realized the potential to extract data from thousands of people. Then, he observed that he could assign tasks using his mother's NIN value.

It's common to see many mobile applications verify users through the National Central Registry System for official processes. We once again witness the importance of keeping in mind the possibility of exploiting this capability.

Therefore, in any officially developed application, unpredictable data like date of birth and place of birth should always be requested during the verification process alongside the NIN to ensure security.

Grasshopper Attack

Payment systems are highly sensitive applications. Even a small overlooked difference directly turns into money cost. This is why we need to meticulously consider every detail when conducting pentests on payment system applications.

Today, one of the most intriguing vulnerabilities we are working on is the race condition vulnerability. While transferring money from one user to another, we're simultaneously sending dozens of requests with turbo features. We notice a discrepancy between the amount of money we sent and the balance.

The relatively challenging emergence and resolution of this vulnerability are due to the multi-server architecture and load balancer structure. When multiple requests come in simultaneously, the harmony within this structure can be disrupted. Therefore, we spend days working with our client on this vulnerability, refining algorithms and architecture until consistency is achieved in all scenarios.

Who Was I, Who Am I Now?

For critical software, self-protection in terms of the TDPL (Turkish Data Protection Law) is of paramount importance. Therefore, some of these software systems retain the IP information of logged-in users.

During our testing on such a critical application, we're contemplating what we can do in that regard. After intercepting outgoing traffic with a proxy, we attempt to send a fake IP value using the X-Forwarded-For header. Surprisingly, the application indeed accepts and displays this fake IP address in its log files.

While this doesn't directly harm the application's functionality, we emphasize the legal significance of this finding, especially in the event of any cyber incident, and report the situation to our client.

Selective Perception

During the location discovery process within the client's organization, we stumble upon an Ethernet outlet situated in an open area but tucked away in a corner. Swiftly and quietly, we test where we can gain access through this port. However, we realize that the port isn't active.

While this outcome is favorable, we still remind our client of the importance of this matter.

Entering Through the Garden Gate

We are conducting a pentest on a beautiful hotel. Their corporate desktop software catches our attention. The application also has API support. When I review the API documentation, I notice that there is no mention of authentication functions whatsoever. Consequently, by sending POST requests via the API, I can modify data like reservations and payments.

It seems that the software company that developed the desktop application solely relied on desktop session login and user authorization. While explaining the situation to the relevant technical team, we also include this finding in our report.

How Secure Are My Boundaries?

We log in from the guest network at the tourist hotel that hosts us. We determine that there is access to the internal network and server block. Within the internal network, we observe that we can reach various locations without proper controls.

We advise our client on the necessity of implementing a more stringent architecture.

Everything Can Be Used As A Trump Card Against You

We notice that there are some Word documents on the client's website. Upon downloading and examining the metadata, we discover that we can gather information about the company's employees.

We also notice that certain PDF files are official documents and have been signed by company authorities. We observe that these materials could be highly beneficial for both personal targeting and scenario development for a “spear phishing” attack. We include all these observations in our report.

Seeing Behind the Curtain

We realize that the target system is positioned behind Cloudflare against cyber-attacks. Despite this, we manage to retrieve a real IP address through a small Censys query.

Further investigation reveals that this IP address belongs to an MQTT protocol server that communicates with IoT devices. We recognize that this machine, which is unprotected, could easily fall victim to a DDoS attack.

Upon detecting this vulnerability, we promptly inform our client.

If the Same Key Is Held by Multiple People

I'm currently conducting a pentest on a small career software developed by our client. The client is quite interested and is observing me live through a projection device. I'm sharing every step I take as if teaching a class.

In the career software, I sign up with a test email account. After logging in, I realize I can change my email address. I wonder what would happen if I entered the email of an existing user in my profile, and I decide to try it out.

Indeed, the system accepts the email of another user as my own. Upon further investigation, we realize that the integrity of the accounts in the system has been compromised. In our analysis, we observe that both passwords for the victim's email are accepted by the system, while the attacking account, despite being registered in the database, is not recognized by the system. It's even possible to sign up again using the same email.

Things Always Forgotten

We are connecting to our client's internal network via VPN. We notice that a specific IP block is associated with IP cameras. Naturally, we first attempt the most common vulnerability we encounter: trying the default username and password. However, we realize that they have all been changed.

Next, we think about one of the most common vulnerabilities in cameras: LFI (Local File Inclusion). We try reading the `http://IP/../../../../../../../../../../../etc/passwd` file. Yet, when accessing it through the browser, it requires a username and password. This time, we use Burp Suite to send a GET request. Indeed, the LFI works, and we read the `passwd` file.

We ponder over how to crack the `passwd` file. john typically cracks the unshadowed version of the `shadow` and `passwd` files. Still, we decide to give it a try and use john. To our surprise, one account is cracked: `test / test1234`.

Extreme Sense of Security

We are pentesting a mobile iOS application. The application also has a messaging feature. Since it's an iPhone, the developers never considered a jailbroken phone issue.

Because our test phone is jailbroken, we put a test.html file in the Downloads directory. Of course, we also embedded a little JavaScript code that will result in XSS. When we sent the file within the app, we saw that XSS was executed successfully. We've also tried many other file types with this excitement, but haven't found any other attack vectors to take it further.

Understanding the Protocol

We are testing the closed network of a very important institution. Our customer wants to make sure that the architecture in his closed network is as secure and hardened as possible. One component of the architecture is a messaging solution based on the Asterix protocol. This is the first time we've heard of this protocol.

Our team is investigating the situation and reveals the message and data format structure from the technical documents of the protocol. Then we listen the network traffic and record it. After that with Scapy, we show that we can generate fake data in accordance with the protofol data format and again, we show that we can enter fake source or destination addresses in accordance with the message format. We enjoy discovering something new.

Again, we are organizing a spoofing attack that generates fake traffic for a VoIP solution, which is among the architectural components. We take a screenshot to show that we have succeed this through the VoIP application's own logs.

My best friend IDOR

We are pentesting the web portal, which is the backbone of the institution. For this purpose, we examine the swagger.json file by importing to Postman to better understand the API functions of the application.

Naturally, our first target is the user profile page. We try to update the profile via API. We discover that the email and phone format control normally which is on the client side is not here. We take a screenshot by entering values such as non-existing-email and non-existing-phonenumber.

We are wondering the effect of the user_id value during the update. We re-update the data by replacing the user_id value of another test user with the existing one. Indeed, we see that we are able to update the profile information of another user, which we normally do not have authorization to do it.

Of course, we immediately change the e-mail of the target person then click the reset password button. The password reset link comes to the e-mail that we entered. Result: account takeover. I understand again why I love the IDOR vulnerability so much.

After some more digging, I understand that the session token does not expire automatically, so the session does not close automatically. More importantly, when I log out and submit old POST request which list products in Burpsuite, I observe that the result is positive. So Authorization Bearer token is not managed properly. Afterwards, I see that the application crashes with a continuous Runtime Error error. Then I realize that the main source of this is that we have already set the default value of 0 in the user role.

Haze the Air

There is a serious difference between a developer and a hacker. The developer focuses on the faultless and performance-based operations of the system or software. The hacker, on the other hand, focuses on the vulnerabilities in that system and how she can exploit them to her benefit. The fundamental difference is one of perception.

When we examine the API features of the web application we test, we see that it has the feature of adding logs manually. We understand that it is possible to produce tens of thousands of fake and misleading logs. We add this vulnerability to our report along with the screenshots. Wolves love hazy air, and log contamination is a method of making the air hazy.

Diving Deep

We meet with our customers after our intercity trip and we start our tests right away in the afternoon. While reconnaissance activities continue, I am also trying an ARP poisoning attack. Although poisoning is possible, I can't find any real password data other than SNMP and SAP Diag findings, which bring public community texts.

Still, I am intrigued by the SAP Diag findings. I notice that many clients are connecting to a server on a completely different network block. Obviously this is SAP Server.

I also feel that there are texts captured from the network packet when I examine it. I even see that the Windows username of the friend trying to connect to SAP is also mentioned. An inside voice says there is more. I'm starting my research. I'm discovering a plugin for analyzing SAP traffic via Wireshark. Only in order to use this plugin, I need to download Wireshark from scratch and recompile it with the plugin. After many failed steps I am finally installing in VMWare Ubuntu environment. But the filter commands I need to see in the plugin are still not recognized for some reason. After a little research, I realize that the plugin is still not defined in the Plugins tab in the Wireshark About menu. I'm copying the sap.so extension to the required path. Yes, finally the SAP Diag plugin works successfully.

However, there is a problem, Ubuntu in the VMware environment does not work in promiscous mode, that is, it does not see the traffic on the actual machine. Despite trying many configurations and settings, for some reason I can't catch the traffic. When I ping the SAP Server from my Windows machine, it does not come, but when I ping it from within Ubuntu, it does. When I ping the SAP Server from my Windows machine, it doesn't reply, but when I ping it from Ubuntu, it replies.

I have no time to lose, I have to be fast. I know that the promiscous mode is much more practical in the virtualbox environment. I reinstall everything in this environment and turn on promiscous mode. I ping from my main machine and bingo now capture from within Ubuntu.

But this is not enough. Because I only see my own main machine. When pinged from other machines, none of them fall on the Wireshark side. I obviously need to do the Man-in-the-Middle Attack. I repeat ARP poisoning and meanwhile listening for traffic on Ubuntu. Indeed, traffic is flowing.

I immediately activate the commands that filter the credentials and start waiting. I do indeed see some small packets drop when the filter is on. When I analyze the packet, I see a readable text value in a tiny field called Expert Info. Bingo, I understand that I have captured the SAP password for that IP user.I am happy to see that my intuition did not deceive me. I share screenshots with our customer, who already follows the process with me.

Killer

We come to our customer’s office to start the pentest. We had already finished the black box step of our testing. Now we are plannig to complete the gray and white box tests. Although we were able to make any changes that we wanted by gaining unauthorized access to the admin management panel of the company's R&D application with zero knowledge during the black box test phase, the technical manager of the company seems to be unaffected somehow. I'm also a little weird and a little angry inside. The admin panel, which normally does not come when logging on to the portal, temporarily appears when we proceed with single steps through Burpsuite, and we discovered that we can manipulate POST requests.

As I begin the test, I decide to try the "killer" first thing. It's almost 20 years old software called "Cain & Abel" that I call the killer. Its work is ARP poisoning, this is old but effective Man-in-the-Middle software that manipulates ARP tables on routers and provide flowing traffic to pass over itself. I am running the application. Normally "Cain & Abel" cannot listen to SSL encrypted traffic. However, it captures passwords which flowing in the IMAP, POP3, http, ftp traffic. I've also witnessed it even capture VOIP phone calls in many cases.

Indeed, less than a minute later passwords being typed on the screen. One of them belongs to the most important e-mail in which the company handles its export business. I am successfully logging into the email with the credentials. Then I invite the technical manager friend and show the e-mail page lively. Afterwards, there was a brief but powerful silence.

History Never Forgets Anything

Our customer is one of Turkey's largest technology companies. They expect us to scan hundreds of open URLs and IPs. They have already had penetration testing service every year for at least ten years.

Although they give us a list, we also use our own discovery methods. One of them is to read the remaining endpoints from the Web archive with *tomnomnom* tools. This way I discover many URL endpoints.

One of them is a small document management application. Also, I can become a member. After I become a member, immediately analyze my own documents through proxy. I am reading a user id value that belongs to me and is associated with the document. After playing with this value a little bit, I see that I can read all the data of other users as well. IDOR is showing up itself again. Moreover, their passwords of hash value seem over there. I am cracking easily one of them and see that it is possible to take over any person's account and I notify the customer without waiting for a report.

Then I continue to examine endpoints. One of them belongs to the "Human Resources" application. I see that I can add a CV that I connected to the URL address. I add it and when I want to delete it again, I encounter a numeric value again. After a little examination, I understand that I can delete all the CVs in the application. I'm reporting this issue to our customer right away.

They are very surprised with this findings. Because we're still in the black box testing phase and they haven't given us a test account yet. At that moment, I realize the interestingness of the situation.

Apparently the endpoint we discovered, allowed me to even get behind authentication. We are preparing an interim report on these findings, without waiting for the final report.

What Do You Share With Whom?

Our customer is a manufacturing company and the IT manager is a very senior and attentive person. Before we start the test, he sizes us upwith his gaze.

We're starting the internal network test. As usual, we proceed from the lowest level of authorization to the top. As the first step, we log in from the guest network. In our scan, we see that we can access the server block, including DCs, in a way we never expected. We note that this is a serious architectural vulnerability. In our scans, we see that all machines are up to date and security patches have been applied. For the first time, we are witnessing a truly flawless network block in this sense.

However, one thing catches our attention. When we scan with a simple tool like Advanced IP Scanner, we observe that many share folders are opened more than necessary. One of them is belong to IT administrator. I start browsing his directories and notice that his VPN certificate is right in front of me. We report the situation immediately**.**

While VPN is a great protocol, as always the weakest point is the human factor. No matter how attentive the IT administrator is, I realize once again how important it is to have an outside eye look at the system against possible blindness.

What Do Servers Serve?

Our client has a very large network surface. Even the discovery phase takes quite a long time. For this reason, we think that focusing on the server block is the most efficient way.

After scanning the server block with Advanced IP Scanner, the pink hooded member of our team decides to focus on Vcenter, which has a web interface and has recently attracted attention with its vulnerabilities. After some investigation, she found that there was a CVE-defined about "File Upload Vulnerability". She immediately sends the relevant packages via Burpsuite and succeed run RCE (remote command execution) on VCenter with root privileges.

Of course, the first thing to do is to combine /etc/passwd and /etc/shadow files and try to crack the password with the john password cracker tool. It seems that the password policy in the product is strong and the password is not cracked even after almost a day. We take screenshots of all these situations and add them to our report.

Developer Self-Confidence

Our customer who visited our office today showed us the IoT product they developed. This is a system that interprets the sensor data received from the user. It also has a web-based panel for tracking processes and operations.

I take the device and give it to our pentest team. Meanwhile, we chat with our guest while drinking tea. The weather is nice and we are in the garden. Meanwhile, we learn that our team can easily enter the management panel with SQL Injection and extract all data in less than half an hour. Our guest is surprised. Later on, we realize that the embedded operating system is not very secure, moreover the source code of the C-based program is inside the device.

Just like the main entrance door of a building, penetration tests represent this in web technologies like that.

Patience Test

We are testing the mobile application of one of the largest financial institutions within the country. This is both a great opportunity and a real test for us, because it is obvious that the application has been tested dozens of times before us. I don't want to finalize this job without finding a good vulnerability.

Fortunately, it's summer time and I can find time to do this. I run this Android application with reverse engineering and start to examine it. The application has SSL Pinning (a mechanism that prevents seeing incoming and outgoing traffic) protection and I want to bypass it. Actually there are tools like Frida and Objection, but somehow I want to choose hard way.

I export the app package, make a small code change and repackage it. After that I generate a fake certificate and sign app with this, and reinstall the app on my phone. But inside the app there is *in_app protection* mechanism, so it is an extra code package that checks for the slightest change in the code or the usage of a fake certificate issue.

I see I have no other choice and become psychopathic. With each step, I find the relevant *smali* code snippet in the app, delete or modify it, recompile it and reinstall to the phone. I repeated this process over 500 times for 12 days without a break. Smali language is actually a frozen language, you cannot get any feedback. To overcome this, I develop my own special methods for the logging mechanism. It really requires a lot of patience. After a while I see that I can completely disable the activation step of the app. And finally I can access my profile page, but some of the pages are quite empty.

On the iPhone side, we see that it is very easy to bypass SSL Pinning with SSL Killchain on a jailbroken phone. Therefore, we can reach our page without any struggle, intercept the traffic and start analyzing. Then we realize that we can change someone else's profile page. This becomes the fun of the day, friends in the organization want us to change the profile picture of other officials with the emblem of the rival soccer team and we do it.

We present our report with these and some other minor findings. They want us to share our findings with the software team. But somehow, the head of the software team acts like an enemy at first and tries to downplay our findings. However, when we talk more, he realizes how much I have comprehensive knowledge about the software code. As he realizes our the effort and serious approach, he starts to appreciate and thank us. Still, I learn a lesson and decide not to meet directly with the software team as much as possible after a pentest.

If there were no difficulties there would be no success; if there were nothing to struggle for, there would be nothing to be achieved.

Are the Doors Really Locked?

The first stage of any pentest is surface exploration. That is, uncovering our customer's scope in the virtual universe. One of the operations in this step, in the called passive information gathering, is DNS Zone Transfer. In other words, listing all sub-URL addresses and corresponding IP addresses, normally closed to the outside world, defined in the DNS servers of the target system.

One of our teammates sees that there is a git sub-domain belonging to the target system and enters it. After some investigation, our teammate discovers there are credentials included in a project belonging to the corporate portal application. Using those credentials, they can log in to the corporate portal.

In this way, once again we see that importance and benefit of vulnerability surface discovery.

Technologies Change, Vulnerability Mindset Does Not

Our customer is a company that develops innovative solutions on serverless architecture on AWS. I am also excited to investigate this new technology. Of course, as usual, I examine the system and its architecture by analyzing all request and response calls one by one.

I'm logged into the system as a ordinary user, and I notice a numeric cookie value, I guess it's a session cookie. I set the cookie value to 1, just as I expected (until I refresh the page) the response now comes with admin privileges.

It is an interesting feeling to experience a very basic scenario in the *Mutillidae* lab, which I always explain in my basic cyber security trainings, in an extreme cloud technology.

Where is Docker Hidden?

We are testing a portal-based solution which is running on-prem. Therefore, it is possible to install the system to our Ubuntu machine and run all kinds of small scenarios.

While wandering around on the portal, I am monitoring whether any operating system command is triggered with the "forkstat - exec,exit" command. Although I see that it is triggered with the backup command, I cannot manipulate it. Because the whole system is running on various docker machines.

When I look at the docker images, I see that one of them is the MongoDB image. After matching the name of the image with 127.0.0.1 in the /etc/hosts file, I try to connect to the database with mongo shell. Just I expected, I connect to the database without asking any username and password, and I understand that I have all kinds of write authorization to the database.

Although this is not a very serious vulnerability in terms of blackbox, I think it is a good finding in terms of tightening.

Chat with Agents

One of the most fun things to pentest is testing enterprise portals that use agents. Whether for machine learning, cybersecurity, health or metric measurement purposes, the hand and foot of portals using agents is naturally none other than small services which running on endpoints.

As usual, after checking for standard vulnerabilities, I set up small scenarios. What can I do if I take over a computer running an agent and I am not the administrator?

I'm detecting the port where the agent and the server connected. Firstly, I am examining the binary file of agent with reverse engineering. Without getting too complicated, I detected a port associated with the local IP address. Then I seize the outgoing traffic and realize that this is indeed the port that interacts with the server.

I connect to the port through the browser and get service information and version as a response. An annoying idea came to my mind. What if I write a simple batch script that tries to connect to that port on the server in an infinite loop? I run it and measure the effect. On the one hand, I follow the process on Process Monitor. When I connect with my browser, directly I get a connection reset error. I stop running the bat file, but the browser connection still persists for a long time.

I try to connect from my cell phone to see if I locked myself, but the problem still there too. While the bat script is running, this time I assign an analysis task to the agent, from the server portal. I see that there is not much disruption in the running of the system. My small DOS attack seems to be fail.

In order to get a deeper understanding of the possible impact of such an attack, I decide to go to the second step to make my lab environment to multi-agent and strengthen the attack payload.

Your Harddisk is My Harddisk

We are at the facility of an important football club in Turkey. I start my scans for pentest. Thanks to Eternalblue, I get a shell on the accountant's computer. It's a pinpoint job.

Then the NAS device catches my attention, 2049 NFS port is open. I measure the mountability of the disk with the show mount command and it seems that it is indeed all mountable. Just like the Metasploitable 2 machine I always give train on.

Then I use the corresponding command to build the disk and start browsing the directories. Which belongs to each user there are protected area with username and password. But after building the disk, I can browse unquestionedly.

I find a file called passwords.txt in one of the user's folders. When I look at its content, I see that it contains many credentials and passwords from the web administration panel to bank accounts. Although we continue the test, I realize that the test is practically over.

What's Flowing on the Network?

As in every pentest, our goal in today's test is take over to Domain Admin authorization too. One of the attack vectors we use under the management of our senior team leader who has sweated in the field of network management for years is NTLM Relay, that is, capturing other machines with the pass-the-hash method by capturing user hashes. However, the Local Admin Hashes that we have captured here cannot log in to other machines due to the effect of endpoint security solutions. At this point, we realize that we have captured a Domain Admin hash.

To overcome this, we perform a relay attack via proxy with the Domain Admin hash we captured and we succeed to get a shell with Domain Admin authorization on the management servers. Moreover, one of these machines is a very important Oracle database server.

The experts who are most successful at finding and catching vulnerabilities in a system are those who have built and managed these systems in the past.

Breaking the Client Side Lock

One of the most difficult environments to protect from vulnerability perspective is the security of portals with multiple and different levels of user roles. Naturally, from a pentest perspective, my goal is to investigate privileges escalation vectors.

I log into the application as both a high and a low authorized user and examine it. I look for IDOR vulnerabilities but I can't find anythink. At one point, I see that the high authorized user can save the analysis results to an environment on the network or cloud, while the low authorized user is not allowed to do and the button for this is frozen.

I immediately cut off the traffic and give the path kali_computer/tmp/dump that I prepared via Burp. Bingo, not surprisingly the analysis results are transferred to my Kali machine.

Tell Me My Rights

When developing a web-based portal, the smooth working of the workflow is naturally at the forefront. Neither the developer nor the user don’t thinks about anything else. However, pentest means being able to think outside of this.

When I log in to the portal with admin privileges, I see that I can set user roles and the corresponding authentication settings. I save this sample traffic with Repeater. Then I log in again with normal user privileges. However, the roles and the corresponding settings do not come naturally in any interface. I still "Send" the sample GET requests again that I saved with Repeater by changing the Authentication Token values on them according to the current user. Indeed, I get the role and user settings page, which should not normally come up. Moreover, in some settings, I see that the credentials are coming as open with passwords and I am shocked.

By completely imagining the user roles and creating a tiny scenario, I successfully managed a kind of privilege escalation attack. Already vulnerability scanning tools only scan the surface, but the real penetration test lies on this fine craftsmanship. In other words the trained person who understands the logic of the vulnerabilities and can apply them in the appropriate location and situation.

Flowing Drops

Web socket traffic is a relatively overlooked protocol in pentests because tiny textual data flows through and manipulating them is relatively unlikely. I'm still looking at web socket traffic for a payment system step by step. After spending some time, I understand that each post is tracked via messageId and logged accordingly.

It is very important that the log of every transaction performed in such sensitive systems is followed in real terms. With this logic, I am monitoring the log status by playing with the messageId value. Just as I predicted, the transaction logs do not drop. Therefore, it is possible to disable the system against our actions. I'm making a note of this to add to the report along with the screenshots.

Don't Tamper With Me

During one of our tests, we're reviewing an agent-based security product. The product has tamper detection feature for self-protection. It generates an alarm to the portal, especially in cases such as a change to the config file or shutting down a service.

I'm inspecting the config file. The portal server has an IP address. I change it and immediately restart the service. Then I follow the situation on the portal side. First it disconnects from the agent and goes into sleep mode. No tamper detection log is dropped in any way. Until hours later, I restored the IP in the config file and restarted the service.

Ignored Backdoor Entry

The site we tested is Wordpress based. As usual, I do my routine scanning with the WPScan tool and its API. I'm detecting existing usernames.

Then I try to attack one of these users via the wp-admin panel. However, on my first try, a commercial security product screen comes up. The fact that the password used was both very simple and wrong obviously triggered the alarm.

I'm thinking, what other door can I find entry through? Then I POST the username and password via XML-RPC. When I do a dictionary attack with this I see that the alarm is not triggered.

Challenge Accepted

Today we are testing a very unique environment: Software-Based Networking, SDN. In the preliminary meetings of this test, the technical project manager kept implying that this environment was very closed and therefore we would not find anything. Already in the first phase of the test, we have locked all the ports available to us behind a router closed from the internal firewall. Weren't we hackers?

Fortunately, after a while, they at least gave us access to the entry-level machines. In the first network and port scan, we see that there is one VNC server. This is a Docker machine and we can login with the default password. We break the root password with the password cracking work we do through the passwd and shadow files, the root password does not surprise us again: 1234.

With these access privileges, we also reach some other Linux-based functional machines. We are sniffing the traffic with the tcpdump tool there. Here we understand the existence of a portal. We access the portal via our web browser with SSH forwarding. It doesn't even ask us for a password. It turns out that we understand that this is the management portal of the entire SDN network.

The more closed and sophisticated the system, the more basic cybersecurity measures are overlooked. Because the Docker image is pre-built and embedded directly into the system. A system developers' first priority is to make the system work as quickly as possible, security is often secondary, especially when time is limited.

Your Proxy Means You

We are testing a corporate portal. The most striking feature of the portal is its ability to delegate power to another member. When I press the menu button about this, I get a warning that I am not authorized. But every once in a while the screen appears and disappears.

This one gets my attention. I cut the same traffic through Burpsuite and go step by step. I understand that there is a user query over the User_ID value, if there is a person, his information is displayed for a very short time. When I query this situation consecutively with the Intruder, I see that I can get a breakdown of the user_ID, first and last names of the personnel in the institution.

Meanwhile, the member of our team, who is also a performing artist, examines the power of attorney even more deeply. When he cuts off the traffic, he realizes that he can change the power of attorney. In this way, he can give power of attorney to another person on behalf of another. In the end, this goes up to changing the password and taking over the account as a result of the fully authorized power of attorney. We call our customer and inform him about the situation without waiting for the report.

Eavesdropping

We examine our customer's Industrial Control Systems (ICS) in the field. We observe that Modbus traffic is flowing through a very sensitive Windows machine. We want to listen to the traffic. The first thing that comes to our mind is to install Wireshark, but we hesitate, especially because it requires a restart on the machine.

In some ICS systems, it is very important that the stem is not interrupted. Our customer is very afraid of breaking any settings and corrupting the working system. Therefore, they do not want us to restart the machine. We also have to develop an alternative solution.

After doing some research, we observe that we can do this directly with Windows' own capabilities:

First of all, we start capturing and recording the traffic with the "netsh trace start capture=yes" command from the command line. But the resulting file extension is .etl. To convert it to pcapng format, we use the etl2pcapng tool developed by MS. Yes, we can now examine the pcapng package in another environment.

In the ICS test, we observe that the most important is the architectural configuration of the ICS networks and components. We see that scenarios where enterprise information systems (EIS) and ICS networks are intertwined, where even TCP-based server systems are quite old and not set up properly, are quite common.

We suggest to our client that the Purdue Enterprise Reference Architecture (PERA) model can be taken as a reference for a robust ICS architecture and these architectural institutions can be customized according to the ICS infrastructure.

Please Trust My Lovely Voice

The pink hooded representative of our team reaches the institution representative by dialing the internal number after an external call.

Hello Ms. Ayşe, take it easy. Can you log in with your password on your computer, most of our staff say they get an error. Can you enter one? Will you get an error?

- Who are you?

* My name is Ayşe, I am new employee at IT department. We are doing a test.

- Ah, welcome. Hold on, let me see. (after a few seconds) I don't seem to have a problem.

* Very nice, but we still decided to change the password of all staff. We will do a general renovation. Can I have your password?

- But does it happen? Anyway, my password is xxx.

The result of our Social Engineering Test: Three out of ten passwords have been successfully obtained.

Give Me A New Entry Ticket

I am making a new web portal pentest. As always, I analyze account settings first. The password reset feature catches my attention. I'm intercepting and inspecting with a proxy. There is a USER_ID value, traffic is tracked with Token. I enter another USER_ID value and see that its password has changed too. More interestingly, I delete the Token value, only the "Token:" statement remains, surprisingly the reset is successful. I understand that I can unconditionally take over an account for which I know the USER_ID value (which is actually not very predictable).

I'm starting to think about how I can get the USER_ID value of other users. I refresh the traffic and analyze it one by one. When I cut off the traffic of the group and roles, I see that the USER_ID value of the roles, including the Administrator, is defined in the Response value. The rest is just gravy.

When I combine this with the password reset, I both get all the credentials of the Administrator account and reset the password. Result: Account Takeover.

Solving the Exam Twice

The web portal where I did the penetration test is an HR software that measures the skills of candidates with prepared exams and ranks them accordingly. For this, an e-mail is sent to the candidate and the candidate starts the exam via that e-mail link.

I wonder what I can manipulate in such a fiction. I'm reviewing the e-mail link. It consists of the id value of the exam and the darkened version of the candidate's e-mail address. First of all, I fill the exam like a normal candidate and reach the result. Meanwhile, I can naturally take screenshots of all the questions prepared for that position.

Then I convert the e-mail value in the incoming link to a meaningless value like 1234 and open it in another browser. Bingo asks me to start the exam process by entering my email address. Now I can complete the exam by answering questions I already know.

The script is now ready; I first apply with a temporary ID and email address. Then, with my manipulation, I complete the exam with my real identity and e-mail address.

I Miss You So Much, Please Make Me an Appointment

We're doing a pretty detailed penetration test of a very large institution. Even at the slightest scan attempt, the system blocks us mercilessly. It is a testing process that really requires patience.

One of the features of such large surfaces is that they leave a lot of traces in the old archival media, which we call the wayback machine. This is how I try out individual endpoints when scanning. My goal is to find an overlooked endpoint. After a while, an address group containing the user ID value in a URL belonging to membership services attracts my attention.

I connect to the endpoint. I am reaching all the information of that member without even entering a username and password. I'm reporting this immediately.

I continue. Well, I wonder how is the situation on the mobile side? Is there a traffic cut as well? I'm even willing to at least discover some API addresses.

In this way, I open the mobile application and see that I can successfully bypass SSL Pinning. Then, the appointment service for corporate customers draws my attention. I'm intercepting the request here with the proxy and making an appointment with a fake id.

Our attack scenario is becoming clear: Fill the entire calendar for a week with appointments made with fake accounts. Users who cannot make an appointment in this way will be greatly dissatisfied.

Of course, in order to avoid such a scenario, I immediately inform our customer of the situation without waiting for the report.

Thinking of the Unexpected, Not the Expected

The portal we tested is very sensitive, as it interacts directly with the citizen and includes payment system features. An experienced member of our team decides to review the feature of creating a subscription to the system. For this purpose, after completing all the fields and completing the membership process, the system receives a message saying "we will contact you after reviewing your application".

However, he clicks on the "I forgot my password" button and after entering the information in the window that opens, he is asked to set a new password. Thus, he sees that he can bypass the approval process and log in directly to the profile page. Being able to see logical gaps is one of the most valuable skills of a good pentester.

In this process, he makes another observation. During the membership process, when he first enters someone else's phone number (although a confirmation code is sent to that number) and then sends a confirmation code to the phone number, he realizes that he can become a member with the other phone and register himself with someone else's phone number.

We immediately report all this work to our customer.

Hey, Who's There?

What if someone has already attempted to attack the system you are testing? What if you come across some clues about it?

A member of our team, who is also a stage performer, notices extra text when he enters the portal interface of an access control system in the environment we are testing. When he examines a little, he realizes that a CVE vulnerability of this portal is being tried to be exploited. Moreover, he realizes that it is possible to log in with the default user name and password in the application, and when the SessionID value is deleted here, the administrator authority can be accessed directly. He understands that even if it is entered without authorization, backup files can be accessed or completely corrupt the system.

We share this situation with our customer immediately. They undertake the next Cyber Incident Response Processes themselves.

If The Door Is Closed, We Enter Through The Window

Our client only wants to do the black box test for his highly valued web portal. He states that he will not give any test user account.

An experienced member of our team starts to examine the portal, and soon finds out that there is a "directory listing" vulnerability in the upper directory of the login page and that he can see the aspx extension files there. He examines each file one by one. He understands that almost all of them, except for some, redirect to the login page.

He continues to explore a little more. He realizes that one of the pages he can access contains a login token. When you log in here, the cookie is also updated in the browser. He tries to reopen all the pages he opened before, and he does indeed.

He is registering a new user from the registration screen. Activate this user account from the administration screen and grant administrator privileges. At the end of the day, the entire system is in our hands.

Cyber Security Destroying Product!

The main purpose of cyber security products is to protect you. However, if the product itself is vulnerable, the opposite can become your weakest point.

The newest member of our team detects an inventory taking and scanning product in the system as part of the pentest. Strangely enough, he notices that the default username and password are turned on in this cybersecurity product. When we enter, we find that everything we are looking for has been presented to us on a silver platter.

We have personally experienced how dangerous it is for security products to be based on the default credential architecture.

Sometimes All You Have To Do Is Pull The Curtain

We are examining a document management system within the scope of Pentest. As always, one of the first places I look is the profile page of my test account. I observe on the page that some properties come by default and I cannot change them. First thing I do is proxy the page and try to play with these features.

However, the POST request that I cut is very complex due to features such as the infrastructure of the portal, ASP.NET and VIEWSTATE. I can't get any results from my experiments. I decide to think about it more simply. I open the inspect page via browser side-page, delete the "disable" tags that make the related properties unchangeable. Now I can play with the features that come by default: I give myself approval authority, I also remove some restrictions. When I hit the save button and go to the profile page. When I come back, I understand that the changes I made are permanent and that the portal only controls the front end.

Many times the weakness is right in front of our eyes, sometimes even shouting like here.

Nothing Happens To Me, My Dear!

The security of the shares can sometimes be overlooked even in the most rigorous institutions. I see that the directory is created for each user on the NAS device of the institution we are testing. When I try to enter, it asks for user / password and I cannot enter. I then run a command that gives an authoritative dump of all directories. I see that there are READ/WRITE permissions in 3 directories and I smile.

Two of them have e-mail archive files with pst extension. I open them and a lot of critical data is displayed on the screen. In particular, I scan e-mails that contain words such as password, pass. Indeed, a lot of data is coming out, albeit a bit old.

It turns out that one of the IT manager friends gave himself full authority. It is none other than his messages that contain passwords from the e-mails I read. The authority and power in the hands of the IT manager can also be his weakest point.

Masquerade

The portal we tested is a career portal, candidates who wish can apply for a job position. In the first stage, a panel requesting the National Id Number (NIN), Name, Surname and Date of Birth pre-checks, if the information is correct and consistent, it moves to the second panel and from there the candidate can enter his personal information.

The equation I set up is simple: I first enter the first panel with a real user data (myself) and move on to the second step. When I fill in the data and put it to proxy, I see that the data from the first step also stops. Just as I anticipated. Afterwards, I am applying with completely fake user id by changing username and other information.

Again, a logical gap and the surface of vulnerability it brings.

Playing Computer Games Inside Systems

The institution we work with knows us well and trusts us very much. That's why I want to do a quick and target-oriented test. In general, the system infrastructure of the institution is quite good. Therefore, I decide to do a "SMB Relay" attack, which is less precautionary, and after a while I see that I have obtained the local administrator hash information of various machines in the server block.

Then again, I can run a command with the username and hash I obtained on the server block, and with this command, I can create a new user on 4 machines and take it to the local administrator group. I connect to these machines via Remote Desktop and take screenshots of them.

I'm trying to turn off Defender on the machine that I connect with remote desktop, but it doesn't allow. I then try to shut it down with the Powershell command and it really succeeds. This means that I can act more comfortably now.

I'm looking at processes and I notice that one of them is a domain user. When I examine a little more with the relevant command, I understand that this user is also in the domain admin group. I open a command line for this user with the token impersonation method. I create a new domain user and take him to the domain admin group. When I want to connect via remote desktop, I see that I cannot connect.

At this point, I want to add a little more color to the environment and (with prior notice) I target one of the existing domain admins and renew the password. With that user, I can connect to the DC via Remote Desktop this time.

I see that I can get the benefit of doing lots of practice and lots of tests by progressing quickly and rapidly.

Hacking Human Perception

We do the Pentest Red Team physical test in a company. Our senior member walks into one of the empty rooms in the corporate office and sits in front of a computer. The computer session has already been logged off, and after a while one of the employees comes.

Our red team member begins to play the scenario: "hello, I just joined the IT team and they sent me here. We're getting alerts from this computer all the time, I came to check."

Employee: "Of course I'll help"

Attacker: "Would you please enter the password?".

Employee: Says "Sure" and enters password.

Attacker: "Would you please tell me the password too, I am going to turn it off and on, so I won't call you every time. If I couldn't fix it I'm going to take it to the tech."

Employee: Tells the password. “Password is ……”

Our team member, after obtaining the password of the computer, browses the files and lingers for a while and takes the computer out of the office, no one asks where you are taking it. The comfort and naturalness of our Red Team member is a major factor in the success of the scenario.

The Power of the Bee Swarm

We're testing a beautifully coded platform that does cloud-based analytics. In general, we try many vulnerability surfaces but to no avail.

Our tester member has the ability to invite other users to the platform. However, this value cannot exceed ten people. Senior member of our team is testing Race Condition and sending a very heavy request packet. As a result, we see that 15 people were sent invitations. We also see that we can go beyond the roles allowed in the invitation and send user invitations with the highest authorized definition.

We're taking screenshots and notes to add this interesting finding to our report.

I'm listening to you wholeheartedly

Our customer wants us to examine the VoIP phones he has purchased recently as part of the penetration test, especially in-depth. Our teammate who does the test is fully involved in this work.

By listening the traffic via Wireshark traffic monitoring tool he gets the traffic flowing through the 5060 ports. Then, when we repeat the one about the call, we realize that we can make the phone ring.

We find out the intercom number of the tea shop room and I I leave the room and go there. A call comes in, moreover, Pentest is written as the caller on the phone screen, spoofing is active.

I pick up the handset and we chat randomly. Then we send packets for DoS attack and the sound on the phone starts to sizzle. Then we listen to the logged speech traffic.

By guessing there is a problem with the codec, we save it for review. We see that we can make partially successful attacks.

I declare you as a Master

We are testing a human resources portal. I see an action menu related to deputize processes. Deputize processes have a complex workflow by their nature. I think I should definitely look here and focus.

I'm creating a deputy form and while I'm delegating to a user my authorization, I cut and review it on the proxy. I wonder what happens if I take the authority instead of giving it, so I change the giver/receiver parameters. I also change the id of the request.

When I send the request, I get a really successful response. Now I see myself as the person who took the authority. While we can't fully verify, this is exactly what an attacker wishes. I think that it can mean that he can get any authority he wants from any person. Moreover, the fact that I can change the id value of the form means that I can break the proxy transactions given by others.

The pentest findings that I enjoy the most are workflow vulnerabilities.

Everything You See Makes You Blind

Everything You See Makes You Blind. If you are focused on only one point, your perspective will narrow over time and you will not be able to look at it from other angles. For this reason, you must constantly change your own memory and reset yourself. Until you discover brand new findings.

I have given a lot of thought to the deputize process before and I'm doing the second phase test of the portal now. I'm thinking what else I should look at.

I'm delegating an authority that I defined earlier to a person for a certain date range. Then I give the same authority with the dates which matches that date range to someone else. The system accepts both and displays them on the screen. Having the same authority at the same time for two people can be very dangerous in many situations.

Exploring in Different Perspectives is an Art

Our client is asking us to test an admin panel login. It will be purely black box testing.

After a little research; On the password renewal screen, I am giving a new password with a random old password. When I try different usernames, I understand that the warning message given when the user is defined and the warning message given when the user is undefined are different. This indicates that there is a user enumeration vulnerability.

Then I do a dictionary attack on a defined user. There is no CAPTCHA protection. But after a while, I get a warning in traffic that is account is locked. I realize that with this administrator account and this method I can always keep this account locked. I also add this to our report as a vulnerability.

I like the artistic aspect of Pentest the most.

Digging Soil for Treasure

We are testing an iOS-based mobile application. After using the application for a while, we explore the iPhone application and examine the files one by one. To see if it contains sensitive data or a file open to manipulation.

After some searching, I noticed that there is a parameter called “ID_Number_Verified” in a file. I'm adding this to our report, because it can be used to create an account with a fake “ID_Number”.

Connecting the Dots

We examine the portals of a company. Only members of that company can log in with their identity information.

Our intern accesses the identity information of a person within the scope of the company through passive information collection methods. I wonder if I can register a member to the company portal with this information? With this information, I try and complete the registration process.

The team leader of our team detects that there is a CSRF vulnerability on the portal at the same time. This situation; means that I can change the data regardless of any authorization process. With the PoC HTML file she prepared, she shows that she can completely change victim information and update it as she wishes.

The result is complete teamwork.

Everything Is Already Before Your Eyes

Everything is before our eyes; all the surfaces of vulnerability are sleeping silently. As long as we know how to look.

In the test environment we are doing, the most experienced member of our team combines traffic poisoning and SMB Relay techniques to capture the hash data of some users. He understands that one of them is the local administrator. When we test this on the machine we selected, we see that we can run commands remotely.

We are coming to a very exciting point. I use this ability to create a new user on that machine and get him to the local administrator group on that machine. Then I can log into the machine with Remote Desktop.

The first thing I do when I log on to the machine is to disable EDR. The scenario I want to try is ready in my mind. I am downloading my Powershell based script package for my scenario. I'm listing the processes on this machine. I see that one of the processes belongs to a real person who is a domain user. The probability of this person being an authorized user on the domain seems quite high.

I am organizing a token impersonation attack on that process. In other words, I steal the user's token in the process and open a new command window with it.

When I type the whoami command, now it shows that I'm that person.

At this point, I create a user on the domain and give him domain admin authority. I get the response that the commands are running successfully.

Now it comes to the most exciting point. I'm connecting to the DC via Remote Desktop and logging in with the user I just defined. Bingo, I'm inside. I open the "Active Directory Users and Computers" tool and take a screenshot.

Golden Fist

Today, programming platforms offer infrastructures that prioritize security. Therefore, the emergence of basic vulnerabilities such as XSS and Sqli has decreased considerably compared to the past. At this point, the basic principle for a pentest that makes a difference; to see the unseen, to think the unthinkable.

I'm testing a new web portal. I'm reviewing the membership system and I'm signing up with a dummy account. At the same time, the e-mail value used for login comes as read-only. My script is clear, can I spoil it?

To update the profile, I proxy it and examine it. While all parameters go into POST request, email parameter is not. It is quite nice. So, I manually add the email parameter and send the POST request.

The result is more interesting than I expected. When I log out of the portal and try to login again, I cannot login. I try the reset password option, it says "user not found". I'm signing up again with the same account and trying to login. I'm still getting the same error notification.

I understand that the operation I have done has affected the entire flow of the system and corrupted the database. I think of Mike Tyson matches that I really enjoy watching. In the first few powerful punches, the opponent is knocked down.

Resetting the Odometer of the Car

I am pentesting a mobile based application. As always, my primary focus is on core features like membership, login and account settings. I also enter my phone number when registering. The application gives me 120 seconds to enter the SMS code I received on my phone and counts down the seconds.

After some time, I press the "Back" button in the browser and press the "Next" button again. Very interesting; The countdown starts again from the 120th second. I realize that this way I can extend the duration of a brute force attack on the SMS code as long as I want.

However, I am still not sure if this only affects the frontend or the API functions in the backend. I'm taking a screenshot to add to my report.

Listen to Your Intuition

Maybe it's because of constantly pentesting, sometimes it's more you feel that there is a problem there without trying the process.

I'm testing a mobile app. I capture and analyze traffic after SSL Pinning bypass. While reviewing the password reset feature, I delete the Authorization Bearer header value and resend the POST request. Indeed, it returns successful results. I see the same applies to the login process. Application session security is not checking strong enough.

I immediately check to the "My Account" section in the settings. If I can change the email value and that is session independent (via the forgot password feature), could I success account hijacking vulnerability?

Unfortunately, I see that my test user does not have this privilege. I still create possible PATH values, by inspecting POST requests. But I am getting errors like "method not allowed" or "unknown path".

As a result, I find this possible situation important and decide to report it.

Moving on the Little-Known Path

Hacking Flutter-based applications through Burp Suite traffic analysis is relatively problematic. After some research I decide to use a new method. This method produces a clone apk that can be tracked by proxy. I complete the process and install the clone apk on my test phone. After intercepting the traffic, I can partially inspect it.

My first focus is on the password reset feature. Here I see that the username is the phone number when I cut off the traffic. Phone number is also the password reset parameter.

This sounds very interesting. Because to renew the account of a victim whose phone number I know, I can forward the confirmation SMS to my own phone number and take over that account. I am reporting this to our customer as a vulnerability that needs to be fixed urgently.

The Pleasure Of Getting In The Ring For The First Time

I think one of the greatest sources of happiness in this world is being able to do what you love. The sooner you decide and name the job you love, the sooner you're on your way.

This is the first test of our youngest team member with us. First of all, she detects a vulnerability in an old version application on an old version machine and gets a meterpreter shell. Then she gains local administrator authority with the rights escalation module. Here she gets the Local Admin hash value from RAM with Kiwi. Then, with this hash value, she sees that she can run commands on many machines in the network block.

Then she examines the processes. She realizes that a process here belongs to a real user. She thinks it should most likely be domain admin. She gets the process token. With this authority, she creates a new user in the domain and takes it to the domain admin group. The next is very easy; Login to DC with remote desktop.

What's This?

Hacking is essentially tampering, basically there is only curiosity. It’s perception always works in this direction.

We are on the way back from lunch at the company where we are working with the Red Team. While our experienced member was going down the floors, network printers caught his attention and immediately started to deal with it.

In one step, he realizes that he can see the list of all users defined on the printer. He fidgets some more and sees that the printer is asking him for a password. The password is so simple that he guesses right at the first try. From here he can now explore the printer management screen.

Later, we also go down to another floor and take a look at the other printers there. While an employee working in that unit was passing by, we ask; "excuse me, we came here to control printers, I'm going to print something for the test, would you please enter the password?". While she enters the password, we seize it too.

As we proceed in this way, the kiosk draws our attention. When our experienced member drags his finger from the corner of the screen, the keyboard screen pops up. From there, he brings up the Windows screen and examines its features. Surprisingly, we notice that there is no Antivirus.

We immediately prepare a meterpreter exe from our attack machine and publish it with python http server. We reach that address from the kiosk machine and download the exe file. We open the command line as local administrator and run the exe. Boom, we have the shell access. Then we steal the token value of the SYSTEM user and then get the NTLM hash value. As I mentioned before, hacking essentially means tampering.

Ammo Guard

I am in the white box testing phase of a web based portal. The account information of the defined users is also included in the panel. Moreover, as this panel is the admin panel, somehow the passwords were written in clear text. Shocking comfort.

Passwords should always be stored with a strong hash algorithm, even in the database. Our assumption should always be: “The enemy has bypassed the guards and crossed the line of positions and is roaming freely inside. The ammo guard must always be on duty and protect the ammos.”

Join And Enjoy The Dinner!

Sometimes a skill we offer users can work against us. For this reason, it is necessary to evaluate the meaning of each ability we offer from the perspective of the attacker.

An experienced member of our team discovers a help page describing API functions in one of the subdirectories while exploring the target portal. When we examine this help page, we are surprised to see that there is no session control.

For example, we see that we can list the contracted companies just with a GET request. More importantly, we see that we can define a new member and approve some financial properties with an appropriate POST request. Moreover, we understand that the Recaptcha feature is also non-functional, so we can send as many packages as we want without being subject to rate limit restrictions, including SMS.

We inform our customers about this without wasting time. Obviously, the help page should be closed first. API functions should only be used by authorized users. Consecutive requests should be actively monitored.

Help Yourself Please.

Sometimes a skill we offer users can work against us. For this reason, it is necessary to evaluate what each talent we offer will mean from the perspective of the attacker.

An experienced member of our team discovers a help page describing API functions in one of the subdirectories while exploring the target portal. When we examine this help page, we are surprised to see that there is no session control.

For example, we observe that we can obtain the list of contracted institutions only with a GET request. More importantly, we find that we can define a new member and approve some financial features with an appropriate POST request. Moreover, we understand that the Recaptcha feature is also non-functional, so we can send as many packets as we want without being subject to the sequential request sending limit (rate limit), including SMS.

We inform our customers about this without wasting any time. Obviously, the help page should be closed first. API functions should only be used by authorized users. Consecutive requests, on the other hand, should actively monitor traffic.

Anthrax In The Envelope

The most crucial aspect of web portal testing is first understanding the purpose and workflow of the portal. Then, combining this with potential vulnerability surfaces and conducting individual tests.

While testing a comprehensive web portal, I notice that it allows users to upload files in both image and document formats. Among the uploadable formats, I see the "eml" (email) file type, which piques my interest.

I attempt to upload an executable file with the ".exe" extension, compressed using zip and rar, but it doesn't allow it. Next, I decide to send myself an email. I attach the rar file containing the executable to the email and send it. Then, I save the email with the attached file in eml format and upload it to the system.

Upon downloading the file from the relevant section, I confirm that a malicious exe embedded in an email (intended to be opened by the recipient) has indeed been uploaded to the system.

A Shared Drive Is More Than Just A Disk

The security of shared directories is often overlooked, yet incredibly significant.

While conducting a penetration test for a large organization, our team's junior member discovers an interesting shared directory. While navigating through the folders, he notice directories composed of user names. Upon closer inspection, he realize that these directories belong to *Citrix* computers. Each disk is nothing but the *Citrix* machine disk of that specific user.

Upon further examination, he find that he can access all the data of that user without actually hacking into these machines. He excitedly realize that he hacked these machines without actually hacking them. Upon closer scrutiny, he observe that some users have been typing passwords for desktops and other environments.

We communicate all these findings to our client in the format of an Emergency Vulnerability Report.

If You're Standing Still, It Means You're Going Backwards

The purpose of cybersecurity devices is to protect a system, but if not handled with care, these systems can become the weakest link in an organization's defense.

While testing a system, our partner team member is scanning the local area network. She notices that there is a SIEM solution in the system – a solution that was set up years ago for a Proof of Concept (PoC) but has never been actively used. As it turns out, this version of the SIEM software has a remote command execution vulnerability. Our colleague exploits the vulnerability and easily takes control of the machine.

Once again, we witness the importance of keeping software up to date, as this incident highlights how crucial it is to maintain software updates continuously.

What If It's Just An Illusion?

As part of our ICS (Industrial Control System) testing, we visit the RMS-A distribution station of the organization we serve. We notice that a facial recognition system has been installed there to control entry and exit. We decide to have a little fun.

We take a photo of a colleague who works there. We then show this photo to the facial recognition system. Interestingly, the system doesn't recognize the face reflected from the phone at all. It's an older machine, but for some reason, the light reflection seems to be more effective than we anticipated.

We contemplate trying this with a printed paper output, but due to time constraints, we end up skipping this scenario.

Constantly Removing The Product From The Shopping Cart

I'm testing a payment system. I've been trying various manipulations with the prices and quantities in the shopping cart, but I haven't been successful. Then, I attempt to remove one of the items from the cart and send this request through a *proxy*.

After clearing the session and cookie information, I resend the POST request. Indeed, I see that the item has been removed from the cart. Yes, a *CSRF* vulnerability exists.

I log out and log back in, then execute the POST request again. However, I notice that the product ID value has changed. Hmm, the solution seems simple: I'll try sequentially deleting the product ID value within a certain range. This way, sooner or later, that product ID will match, and the item will be deleted.

I try this solution, and I indeed see that the cart has been emptied. I realize that an attacker could render the e-commerce site completely non-functional by periodically sending this POST request continuously.

How Securely Are Security Solutions Configured?

Many organizations invest significant resources in security solutions and implement tightening measures as much as possible. The problem lies in the fact that if these solutions aren't properly configured and integrated with other solutions, serious vulnerabilities that go unnoticed can emerge. The primary goal of our Red Team service is to identify these vulnerabilities and contribute to their prevention as much as possible.

On a corporate test machine belonging to our client, I attempt to gain shell access to my external *C&C* server using a obfuscated *PowerShell* code. Unexpectedly, I succeed in doing so.

However, they already have robust security solutions in place. We immediately review this situation together and address the configuration gaps by analyzing and closing the escape points.

In the meantime, I decide to develop an automated code that obfuscates any *PowerShell* script in a way that won't be detected by antivirus programs.

Being able to write original code with just a touch of creativity is more effective and powerful than one might think.

How to Purchase Team Specific Ticket Series?

As usual, we're conducting tests for a client, an organization that is particularly cautious about not granting us access to their internal networks. They want us to explore the possibilities from their guest network to the fullest.

We have a domain user account with no privileges and access to the guest Wi-Fi network. We begin roaming inside the office and notice an IP phone in a meeting room. We quickly grab the computer and move to that room, where we plug our computer into the IP phone's network cable.

We start our tests and realize that the phone's traffic is interceptable. From this point, we wonder how far we can go. We see an available network port on a wall jack and plug in our network cable, obtaining an IP address. We then start experimenting with the least privileged user we have, who possesses local user rights on the domain.

With our computer, which is not part of the domain, we examine the vulnerable certificate templates on the certificate server from our own computer. Indeed, we identify a vulnerable template.

Subsequently, by exploiting this certificate, we manage to obtain the hash value of the password for the domain admin user. We prepare the command that allows us to log into the domain controller with this hash value and gain access to the network.

Challenge Accepted

One of the most enjoyable aspects of the Red Team service is pushing your limits and doing so with creative methods whenever possible.

I present a fun challenge to the young members of the team: "Send a targeted phishing attack to my corporate email account. The email should bypass the email gateway and arrive in my inbox without being flagged as spam. When I open it, your prepared keylogger should run without being detected by any EDR or filters, and the keystrokes I type should reach your C&C server. Whoever accomplishes this will be treated to a hearty meal."

In the middle of the night, the youngest member of the team sends me his email (with prior notice). His custom-made keylogger works successfully in his own lab environment. However, the email he sent doesn't reach me because the email sandbox blocks the attached file. He then sends a link via Google Drive. The link arrives, but when I click on it, the EDR immediately comes into play.

Undeterred, our team member decides to dig deeper. The challenge and the promise of a meal continue.

Don't Be Fooled By Every Well-Dressed Person

The MITRE ICS Matrix outlines the potential steps of a sophisticated cyber attack on ICS (Industrial Control Systems) systems. As part of the architectural analysis of the ongoing ICS test, we are utilizing this matrix.

Given our concern about the risks associated with USB inputs, I decide to test a Bad USB that we've prepared on the operator's computer. This Bad USB presents itself to the system as a keyboard, so the system does indeed detect it. However, our IT administrator friend has taken precautions, ensuring that the application takes over the entire screen and prevents any other actions from being reflected. Furthermore, they have even disabled command-line and PowerShell-based commands. As a result, we are unable to measure the reactions of the script we wrote. I commend our friend for their foresight.

Nevertheless, I still assign myself the task of exploring alternative ways to carry out concrete and potentially dangerous actions in the background, wondering if there could be another avenue to pursue.

How Can I Become a Member?

In the penetration tests we conduct, our fundamental goal is to astonish our clients with our findings and, if possible, even shock them. Penetration testing is, in many ways, an art form for us. What is expected of us is to uncover something unprecedented and previously unnoticed in a system that has already been scrutinized multiple times. Oddly enough, after a while, our instincts become so harmonious that we can feel the presence of weak points in a system almost at first glance.

I open the portal we are testing. The login screen appears. I glance at the URL bar, and even though it's not part of the interface, an impulse to type "register" instead of "login" comes over me without any prior scanning. Indeed, I discover that I can register for the portal. I quickly sign up and log in with this account. After a bit of exploration, I realize that I've logged in with super admin privileges and have complete control, including user accounts.

In a way, I find it a bit perplexing. In a system that has been reviewed countless times before our arrival, why hasn't a vulnerability that could be exposed with a simple "dirb" scan come to light? I see the danger of a careless penetration test creating a false sense of security within a company, and it's actually a bit disheartening to witness.

The Mystery of Discovery

Cybersecurity and hacking are all about capturing intriguing gaps or relationships in subtle details. It's more of an art than pure technical skill. It's an art combined with a feeling, and that feeling is "mystery."

I am working together with a friend who is a skilled programmer to conduct a penetration test on a "Business Automation Software" that has been live in the field for a while. The software, which is a Single-Page Application, naturally relies on the capabilities of JavaScript. As a frontend-focused application, it operates efficiently and swiftly, and many of the classic web vulnerabilities are fundamentally mitigated. However, one problem I have observed repeatedly in such architectures is the poor resistance to privilege escalation vulnerabilities. A request that does not come through the router may be forgotten.

I start by logging in as a business owner and explore the functionalities of the application. The role assignment functions for other users catch my attention, so I take a closer look at them. Next, I log in with intern privileges. With the help of my friend, we write a *JavaScript* function in the browser console that assigns the user's role. Indeed, our intern user gains instant access to the business owner's screen as the function executes. From there on, complete control is in your hands.

My friend is thrilled with the discovery we've made during this joint effort. As for me, I once again realize how impactful it is to achieve a result by typing the right commands into the browser screen.

First Allow, Then Audit

Catching and exploiting a vulnerability has an artistic aspect to it. Follow the flow, understand, discover the manipulation point, progress, and exploit. While examining a web portal, I notice that some dealers have managed to bypass a part of the approval mechanism required from higher-ups. My client asks for assistance in this matter, and their faith in me and my team's abilities brings me great satisfaction.

I study the workflow of the software and proceed until I reach the approval step. As I always do, I send the action to the *proxy* when I click the button, preserving each step for deeper analysis.

Upon inspecting the response of the initial POST request, I realize that the screen for the next authorized step appears actually. However, in subsequent steps, an audit mechanism kicks in at a certain point and redirects the user back to the starting point.

At this point, I repeat the process. However, this time I "drop" all the steps following the initial one and proceed. After a while, the audit mechanism is also bypassed, allowing the approval process to complete.

A sequence flaw in the workflow, imperceptible to the normal eye, gradually reveals itself as I delve deeper, granting me the opportunity for privilege escalation.

The Joy of Teamwork

We have been testing a very comprehensive and complex portal for some time. The software's workflow is so intricate that uncovering the vulnerability surface is proving to be quite challenging. Moreover, the testing steps progress slowly and are relatively mundane due to our VPN connection.

In this situation, we decide it's best to turn the work into fun. We gather with our most experienced colleague online and decide to approach the test as a team effort. Since it's already evening, we also invite available colleagues to join the party if they wish.

The rest unfolds like solving a puzzle. At each step, we decide what to try and proceed accordingly. Eventually, we manage to bypass the control mechanism that had eluded us for some time and successfully manipulate the system. The live observation of our approach also helps broaden the horizons of our less-experienced colleagues.

We even invite our client, who is very interested in cybersecurity, to join. They participate and we immediately explain our approach to them.

Personally, despite all the fatigue, I feel the joy of being part of a team of individuals who have a passion for their work and enjoy what they do.

While Technology Renews, Fundamental Principles Remain Unchanged

The JSON format is the most practical way to define data in a structured, object-oriented, and flexible manner. Its emergence as a replacement for XML is not a coincidence. This transition extends to NoSQL databases as database extensions and API infrastructures as web extensions, now encompassing the most up-to-date solutions.

A side effect of this evolution is mobile security. Many mobile-based applications now primarily interact with web API functions, essentially becoming clients interacting with web APIs. Naturally, mobile security and web application security have become increasingly intertwined.

However, mobile applications still have their vulnerabilities. I examine an application on a rooted phone during testing. I delve into all the caching, temporary files, and databases left by the application while running on the device. A couple of database files catch my attention. I extract them from the device and use an SQL reader to examine their contents. An ID value representing the user's identity draws my attention. I modify it and *push* it back to the same location on the device.

I open the application on my test phone and astonishingly find myself inside with someone else's account. Another instance of Insecure Direct Object Reference (IDOR), but in an entirely different environment and platform.

Relying solely on client-side controls is always a critical mistake. While technology and platforms may evolve, fundamental principles and approaches remain unchanged.

Even If You're A Human, Sometimes You Have To Think Like A Machine

We are examining the online presence of our client. After a specific surface discovery scan, we come across the presence of a Swagger documentation page that reflects their API queries. Without delay, I execute the queries on the page to understand where an anonymous user can access.

I assume that most of the data returned in response to these queries is not sensitive. However, I notice that one query allows access to certain analysis reports intended for customers.

When I run the query through the API, I observe a long piece of data in Base64 format. Believing it could be a PDF file, I turn to an online Base64 to PDF converter. After performing the conversion, I confirm that a PDF document is generated, and it turns out to be an actual analysis report.

Recognizing the potential sensitivity of this situation, I decide to report it for further examination.

CONCLUSION

The purpose of this work is to introduce the world of penetration testing to non-technical individuals in a simplified manner, often breaking down complex concepts. It aims to familiarize anyone who codes, develops software, manages systems, holds administrative roles, or has an interest in the field with this mysterious realm.

As seen from the logs, in addition to technical knowledge, the perspective on systems and events, as well as the approach to subjects, are crucial components of the penetration testing process.

Scans relying solely on automated vulnerability assessment tools are, as the name suggests, vulnerability scans - they do not constitute a true penetration test.

We hope that this work will help solidify this crucial detail in the realm of penetration testing. Thank you for joining us in this enjoyable journey.

ABOUT US

Bilishim Cyber Security and Artificial Intelligence Inc. was founded in 2018 at Hacettepe Technopolis with the vision of developing innovative Cyber Security solutions using Machine Learning techniques.

Today, Bilishim Cyber Security is a penetration testing company with a TSI-A certification and is one of the few companies authorized by EMRA (Energy Market Regulatory Authority) to perform penetration tests in Industrial Control Systems (ICS).

Among its references there are some of Turkiye's most significant institutions and companies. Bilishim has also developed a Corporate Vulnerability Scanning Tool named Nettarsier and a mobile application focused on fighting against fraud named UMAY.

Our team, known for its Red Team services, offers comprehensive solutions as a package to the organizations it serves, based on its ongoing research and development efforts in this field. We transfuse our experience and knowledge we gained on the field to a software called Red Katana.