

Ethical hacking summative assignment

**Hussein Technical University**

Abdullah Istanbuli | (21110366)

Network Security

**Table of Contents**

**Packet sniffing ……………….…………………………………………………………………………………3**

**Password cracking ………………………………………………………………………. ………………. .4**

**Social engineering ……………………………………………………………………. …………..…...…..4**

**Countermeasures ……………………………………………………………………………………….…...5**

**Web architecture ……………………………………………………………………………………………..5**

**common web vulnerabilities …………………………………………………………..……………… 6**

**exploitation ………………………………………………………………………………………………….…6**

**counter measures ………………………………………………………..………………………………….7**

**passive recon ……………………………………………………………………………………....…………...7**

**shodan .………………… ………………………………………………………………………………………... 8**

**nsllokup ………….…………………………………………………………………………………………………9**

**whois ………………………………………………………………………....………………………………..11**

**nmap …………………………………………….…………………………………………………..……...….14**

**nesus …………………………………………….………………………………………………...……… 16-40**

**wireshark telnet …………………………………………….…………………………………..……...….41**

**wireshark FTP …………………………………………….………………………………..……...….42**

**hydra …………………………………………….………………………………………………..……...…. 43**

**sql injection …………………………………………….……………………………………..……...….** **44**

**XSS …………………………………………….……………………………………………..……...….** **44-45**

**SQL map …………………………………………….……………………………………………..……...….** **45**

**commaned injection vunrabilies ………………………………………………………..……...…. 47**

**manual sql …………………………………………….………………………………………..……...….** **48**

**top 10 …………………………………………….……………………………………………..……...….49**

**Metasploit …………………………………………….………………………………………..……...….51-58**

**CTF …………………………………………….……………………………………..……...….59-63**

**References …………………………………………………………………………………………………..……………………………….63**

1. **Packet sniffing**

Packet sniffing is the process of monitoring and capturing all data packets passing through a given network using a software application or hardware device, it allows the attacker to control and access all traffic from the network, for example, if an attacker was able to perform a successful packet sniffing on your network the attacker could steal the credentials for your emails and social media platforms or capture your bank account credentials.

How can a sniffer be set:

in a shared ethernet, all devices use the same connection and share bandwidth when one device sends data to another all devices receive the data but when devices match the attended receiver and do not match the devices drop the packet but a device running a sniffer wouldn’t drop the packet and instead capture it, a sniffer in shared ethernet environment is passive and hard to detect

in a switched ethernet environment the hosts connect with a switch instead of a hub, the switch maintains a table that keeps each computer's MAC address and port and delivers the packet to the intended receiver, in this case putting a machine NIC into promiscuous mode does not work, so many people think that using a switch secure and immune the network to sniffing which is not true

ARP spoofing: a machine can send an ARP reply without receiving a request, and it can accept a reply, when an attacker wants to sniff the traffic it can ARP spoof the gateway of the network (usually a router) the targeted machine will have an incorrect entry for the gateway which allow all traffic to destined to the gateway to go through to the attacker.

MAC flooding:

As we mentioned earlier switches keep track of devices MAC addresses and which ports they are connected to however switches have limited memory MAC flooding uses the limited memory, an attacker can flood the switch with multiple entries of fake MAC addresses until the memory of the switch is full and the switch enters fail-open mode, where it starts acting like a hub and broadcasting packets to all ports, once this happens the attacker can use the method explained above to perform sniffing attack.

There are two types of sniffing attacks passive and active, passive as we explained in the first two paragraphs, and active using ARP spoofing and MAC flooding or other methods like DHCP attacks, DNS poisoning, and others.

What protocols are vulnerable to sniffing?

Any protocol where the packets aren’t being encrypted is vulnerable such as:

telnet, Rlogin, HTTP, POP, IMAP, SMTP and NNTP, FTP, are all uses clear text to send some or all data which leaves them vulnerable

Is there a way to detect a sniffing attack?

Yes, there are some ways where you can detect a sniffing attack:

Check the devices running in promiscuous mode, and run IDS or network tool that monitors traffic for unusual activity

1. **Password cracking**

Is the process of recovering passwords by the data transmitted or stored by a computer

The purpose can be to recover forgotten passwords or gaining access to a computer

It’s the first step in hacking attempts usually because of its value this can be done manually by guessing it or by using automated tools such as a dictionary or brute-force method.

Types of password attacks:

* Non-electronic attacks: this happens when a person tries to find your password by looking over your shoulder or going through your garbage or doing social engineering
* Active online attacks: this happens when the attacker directly communicates with the victim’s machine and this is done through dictionary, brute forcing, hash injection, trojan, spyware
* Passive online attacks: this happens when the attacker doesn't communicate with the victim’s machine and this is done with wire sniffing, man-in-the-middle, or replay attack
* Offline attacks: this is done when the attacker takes a copy of the password file and tries to crack it offline

1. **Social engineering**

It is the art of manipulating people to reveal sensitive information to use it to perform some malicious action, most often employees are not even aware of a security lapse on their part, only educating can minimize an attacker's chances

Factors:

* Insufficient security training
* Unregulated access to information
* Several organizational units
* Lack of security policies

There are many ways to do social engineering it can be done human-based like dumpster diving, baiting, and quid pro quo, impersonation, or by a computer-based like phishing, or spam email, or it can be done mobile-based malware, using fake apps, SMS phishing.

1. **Countermeasures for those attacks**
2. Packet sniffing:

* Data encryption: use protocols like HTTPs, SSH, and VPNs to ensure traffic is encrypted.
* Network segmentation: use VLANs to segregate sensitive traffic and reduce the risk of exposure
* ARP spoofing prevention: use static ARP entries for critical devices like getaways, and might deploy spoofing detection tools like ARPWatch
* MAC flooding prevention: enabling port security on switches to restrict the number of MAC addresses per port
* Network monitoring: deploy IPS, IDS to monitor for unusual network activity

1. Password cracking:

* Strong password policy: require a long and complex password with mixes of characters upper lower case, numbers, and special characters
* Multi-factor authentication: adding this layer is very important and makes it very hard to carack into on of the accounts without hacking your phone as well so the job would be twice as hard
* Limit login attempts: limiting login attempts would prevent both dictionary scripting and brute forcing
* Secure communication: when the communication between devices is secure the attacker cant use a sniffer to steal credentials or sensitive data
* Don’t write down your password: a lot of people makes the mistake of writing their password on a paper and hiding it which make you vunruble to many attacks

1. Social engineering:

* Implement security policies
* Email security
* Secure physical access
* Having employees courses for awareness

1. **Web architecture**
2. **Role of http https**

A web server is a computer system that stores, processes and delivers web pages to global clients via the hypertext transfer protocol (HTTP).

When someone wants to access anything on the internet the browser generates a HTTP request. The web server takes the requested content from a server and replies with the HTTP response with the requested content

However HTTP lacks security as the process of data transfer is done with plain text (not encrypted)

To address this HTTPs were introduced which work similarly to the HTTP but the transfer of data is encrypted ensuring CIA security

1. **Common web vulnerabilities**

* **SQL injection**

a code injection technique where the attackers enter a SQL query in input fields exploiting the web application-database communication this can lead to unauthorized access, data theft, data manipulation

* **Cross-site scripting**

A vulnerability that allows attackers to inject malicious script into web pages vied by other users, this can steal cookies, sessions tokens, or sensitive user data

* **Session hijacking**

Where an attacker gains unauthorized access to user session by stealing their session ID, when that happens for the website the attacker appears to be the user

* **Dos/DDoS attack**

A type of attack where a server or any type of network traffic receiver gets overwhelmed with traffic rendering it unavailable to users, DDoS involves multiple systems to amplify the attack for example a botnet

* **Cache poisoning**

When an attacker injects false information into the cashing system, that would lead to users being redirected to malicious sites

* **Man in the middle**

When an attacker puts himself between the user and a desired service without their knowledge this makes any data vulnerable to alteration

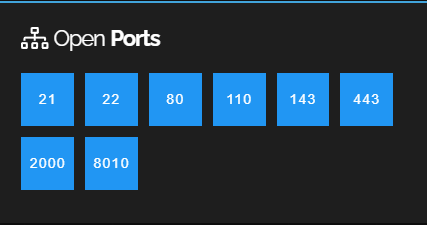
1. **Exploitation**
2. SQL Injection – An attacker can exploit an input field to insert malicious SQL code. For example, by entering ' OR '1'='1 in a login field, they can bypass authentication and gain access to the system.
3. XSS – Attackers can inject malicious JavaScript into web pages that execute in the context of other users' browsers. This can steal cookies, leading to session hijacking or data exfiltration.
4. Session Hijacking – Exploited by stealing session cookies through XSS or using network sniffing tools on unsecured connections (e.g., HTTP instead of HTTPS).
5. DoS/DDoS Attack – Attackers send massive amounts of traffic to exhaust server resources, making the service unavailable.
6. Cache Poisoning – The attacker exploits vulnerabilities in caching mechanisms to serve users altered content, redirecting them to malicious sites.
7. MitM Attack – Exploited by intercepting and modifying communication in real-time. For example, attackers on the same public Wi-Fi network can intercept login credentials if HTTPS is not enforced.
8. **Conter measure**
9. SQL Injection – Use parameterized queries, prepared statements, and input validation to prevent SQL injection attacks.
10. Cross-Site Scripting (XSS) – Sanitize user inputs, encode outputs, and implement a Content Security Policy (CSP) to block malicious scripts.
11. Session Hijacking – Use HTTPS for all communication, set secure and HttpOnly flags on cookies, and implement session expiration and regeneration.
12. DoS/DDoS Attack – Use rate limiting, load balancing, and DDoS mitigation services (e.g., Cloudflare, AWS Shield) to handle high traffic loads.
13. Cache Poisoning – Validate and authenticate cached content, and use secure cache mechanisms to prevent poisoning.
14. Man-in-the-Middle (MitM) Attack – Enforce HTTPS using strong TLS configurations, and implement certificate pinning and end-to-end encryption.

**Part3 Reconnaissance and vulnerability analysis phase**

1. Passive recon/collecting data

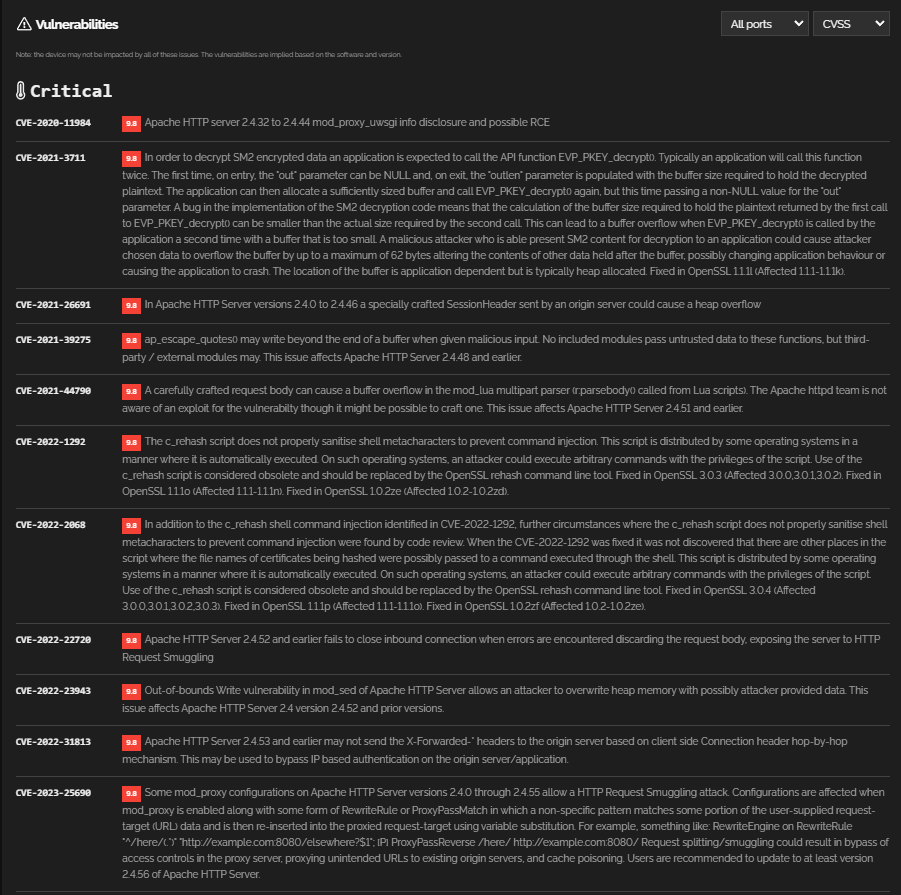
Using shodan.io

Using this tool to scan the targeted website we can see all open ports in the website which can each lead to a collection of vulnerabilities



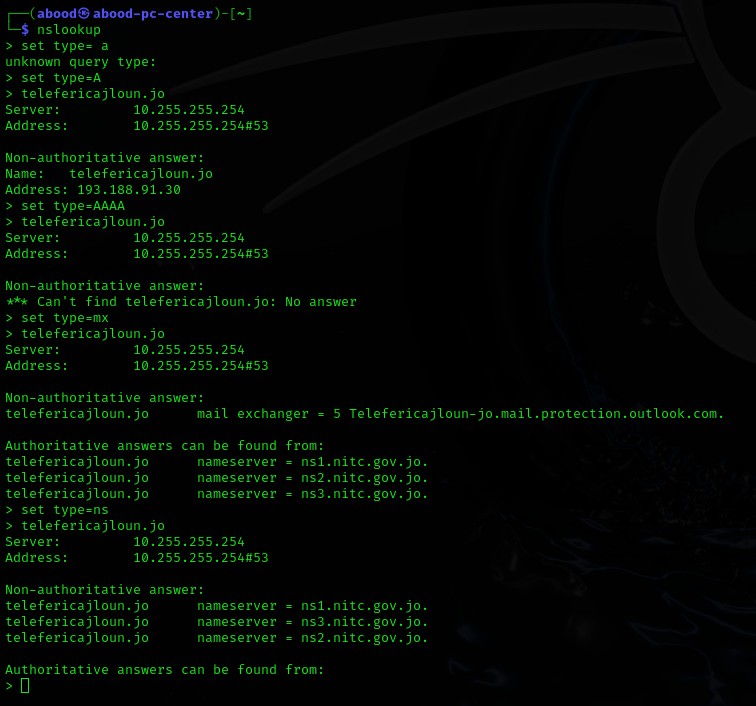
The image showcases the outcome of a Shodan scan on a targeted website. This step was the first of passive reconnaissance phase that we will go through, aimed at identifying open ports on the website. The scan results help in assessing potential entry points that an attacker could exploit.

As well as revealing a collection of vulnerabilities from weak, moderate, and severe in severity



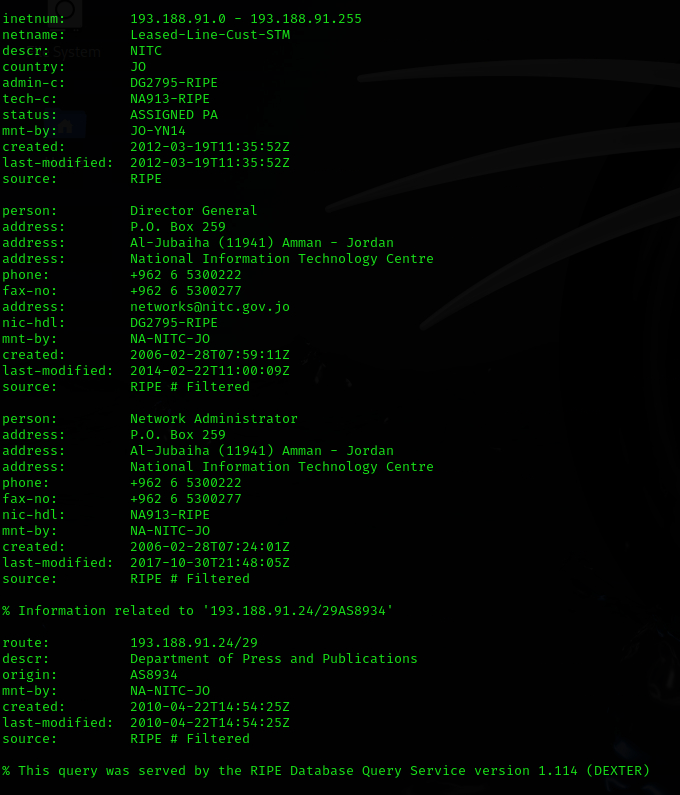
This image presents the vulnerabilities detected on the previous ports, using Shodan. The identified vulnerabilities are categorized by severity, providing critical insights into possible weaknesses that could be exploited by attackers.

Now we try to use the tool nslookup to find dns server’s ip address



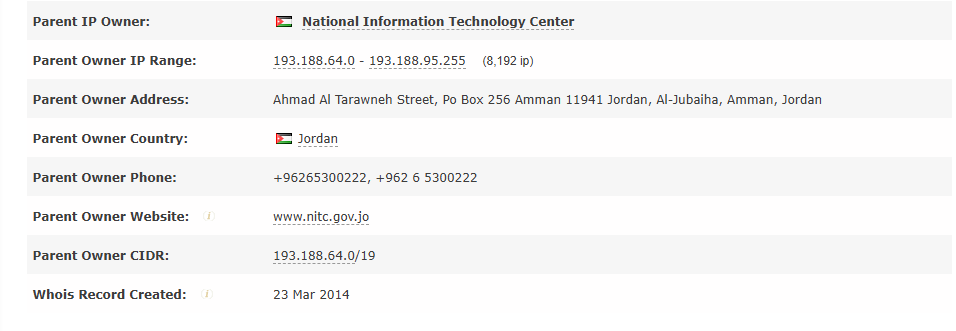
This image displays the results of running the nslookup command with various record types, including A, AAAA, and MX. This information reveals the IP addresses and mail exchange servers associated with the targeted domain, assisting in further reconnaissance and vulnerability analysis

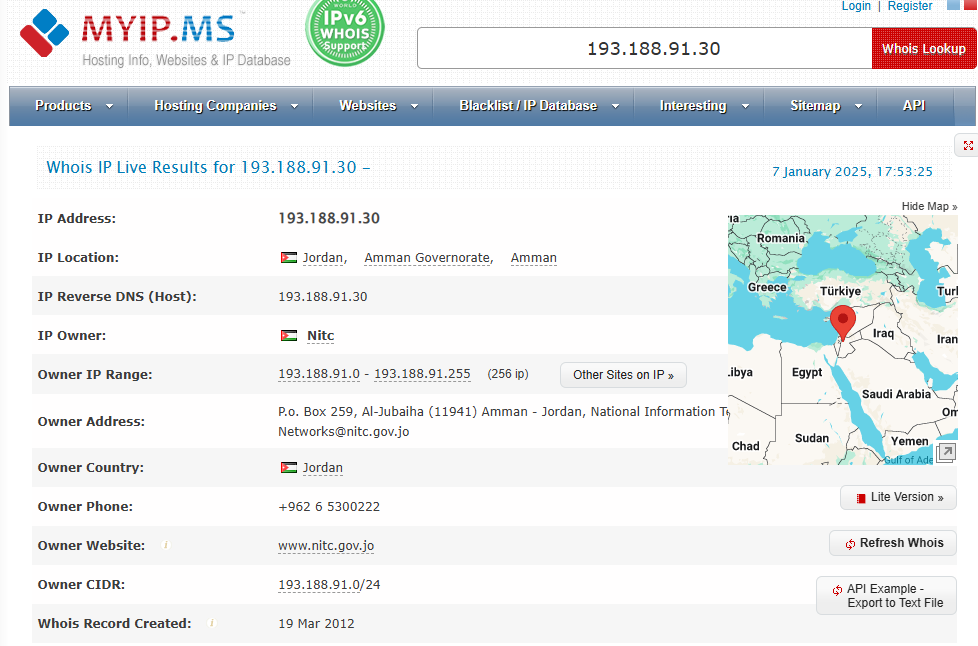
After that we can take a look using the tool whois to find some other general information about the website targeted

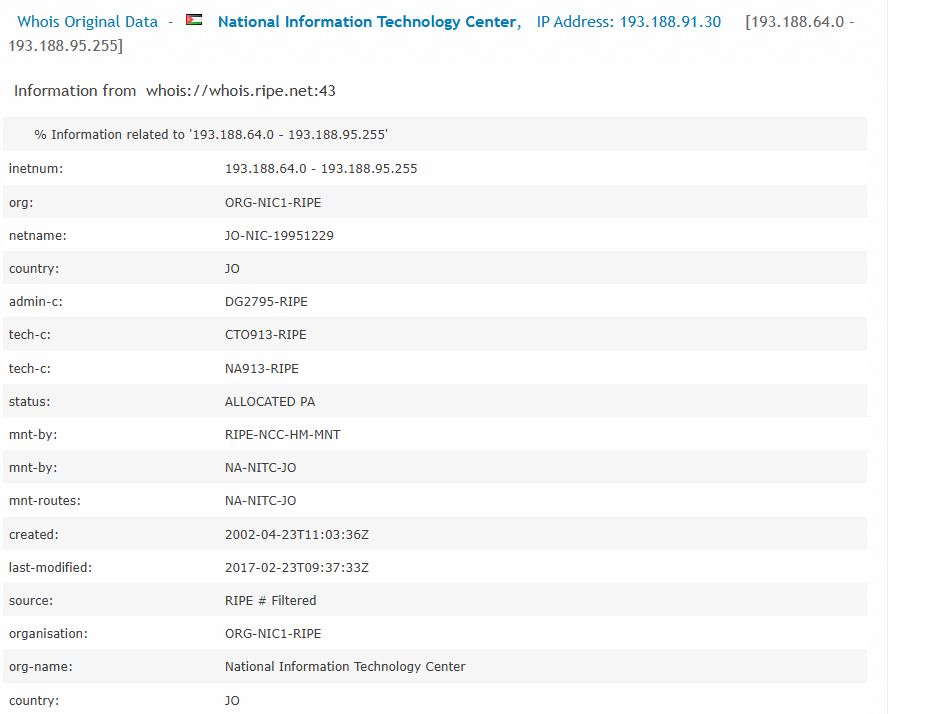


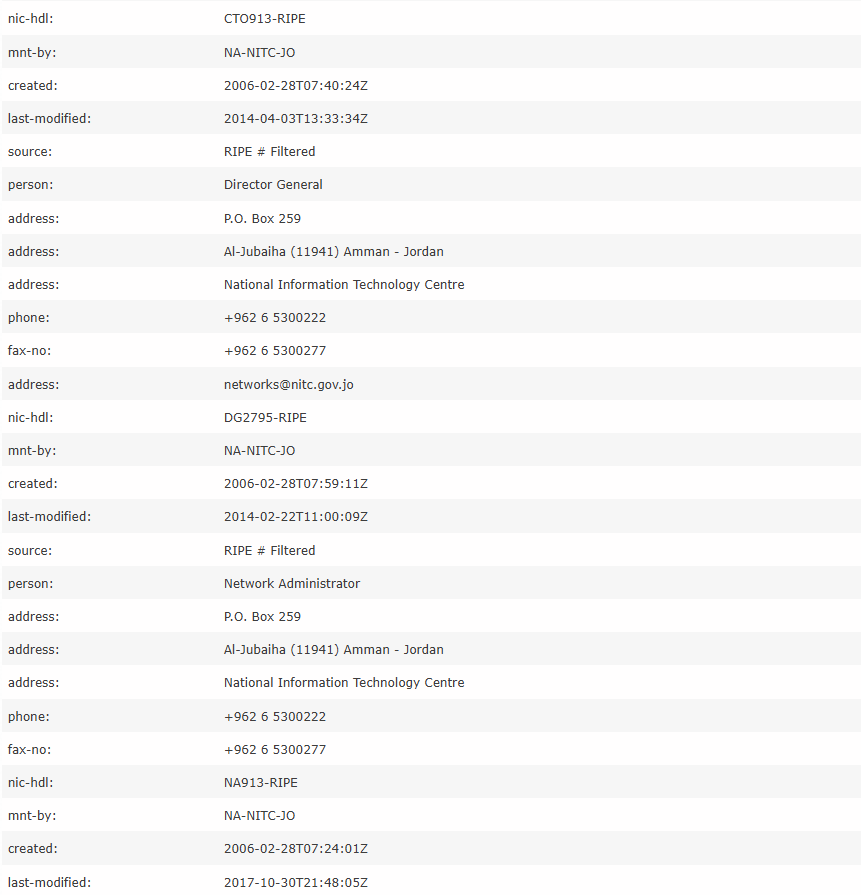
This image shows the result of running a WHOIS query on the targeted website. The output includes domain registration details, nameservers, and other administrative information useful for gathering general intelligence about the target.

Now we can use another website that uses previous recorded of whois







****

****

The final set of images shows the output from using the web tool myip.ms to gather additional intelligence on the targeted website. This tool provides detailed information, including hosting details, IP ownership, and other relevant data that can aid in mapping the website's infrastructure.

**3.B.**

|  |  |
| --- | --- |
| Requirement: | SAMI-CYBERLAST |
| Ip address: |  |
| Open ports: |  |
| Operating system: |  |
| Port 21: |  |
| Port 135: |  |
| Port 139: |  |
| Port 443 |  |
| Port 445 |  |
| Port 3306 |  |
| Port 3389 |  |
| Ports:  49152  49153  49154  49155  49156  49157  49158 |  |

Part 3 C:

|  |  |  |
| --- | --- | --- |
| Issue version | | Issue details with solution |
| Apache 2.4.47 | |  |
| Apache 2.4.7 | |  |
| Apache 2.4.60 | |  |
| Apache 2.4.55 | |  |
| Apache 2.4.49 | |  |
| Apache 2.4.54 | |  |
| Apache 2.4.56 | |  |
| Apache 2.4.52 | |  |
| SSL Certificate using weak hashing |  | |
| PHP 5.6.11 Unsupported | |  |
| PHP 5.6.18 | |  |
| PHP 5.6.19 | |  |
| PHP 5.6.20 | |  |
| PHP 5.6.21 | |  |
| PHP 5.6.23 | |  |
| PHP 5.6.24 | |  |
| PHP 5.6.25 | |  |
| PHP 5.6.26 | |  |
| PHP 5.6.29 | |  |
| PHP 5.6.31 | |  |
| PHP 5.6.32 | |  |
| PHP 5.6.34 | |  |
| PHP 5.6.40 | |  |
| PHP 7.1.31 | |  |

The vulnerability assessment conducted using Nessus revealed a significant number of critical, high, and medium-severity vulnerabilities affecting the target systems. The identified vulnerabilities include, but are not limited to:

* PHP Remote Code Execution
* Stack Buffer Overflow
* PHP Unsupported Version Detection
* SSL Certificate Signed Using Weak Hashing Algorithm
* Apache Authentication Bypass
* Apache Forward Proxy Denial of Service (DoS)

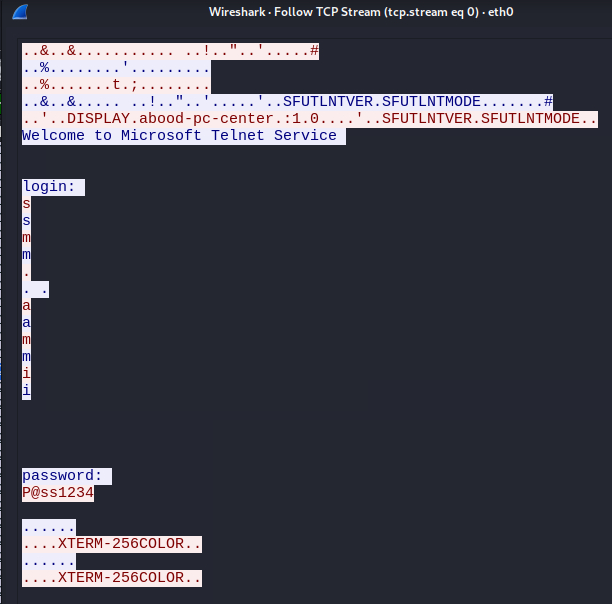
Most of the vulnerabilities are attributed to outdated or unsupported software versions, improper configuration, and the use of weak cryptographic algorithms. The primary recommendation for mitigating these issues is to implement timely updates and patches to affected software components, such as PHP, Apache, and SSL certificates. Additionally, ensuring the use of secure configurations and regularly scheduled vulnerability scans is critical for maintaining a secure environment.

It is important to establish a robust patch management process and maintain up-to-date software versions to reduce the risk of exploitation by potential attackers.

**Part 4**

1. Using wireshark to capture packets during telnet and FTP

1.telnet:

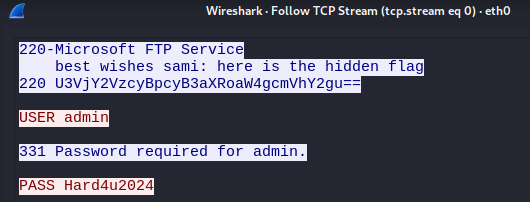


The use of Telnet for remote management poses a serious security risk, as it allows attackers to perform credential theft, unauthorized access, and potentially compromise the entire system. This vulnerability significantly increases the risk of man-in-the-middle (MitM) attacks and network reconnaissance activities by adversaries.

**Recommendation**

Disabling telnet and using SSH which provides encypted communication and secure authentication

2. FTP:

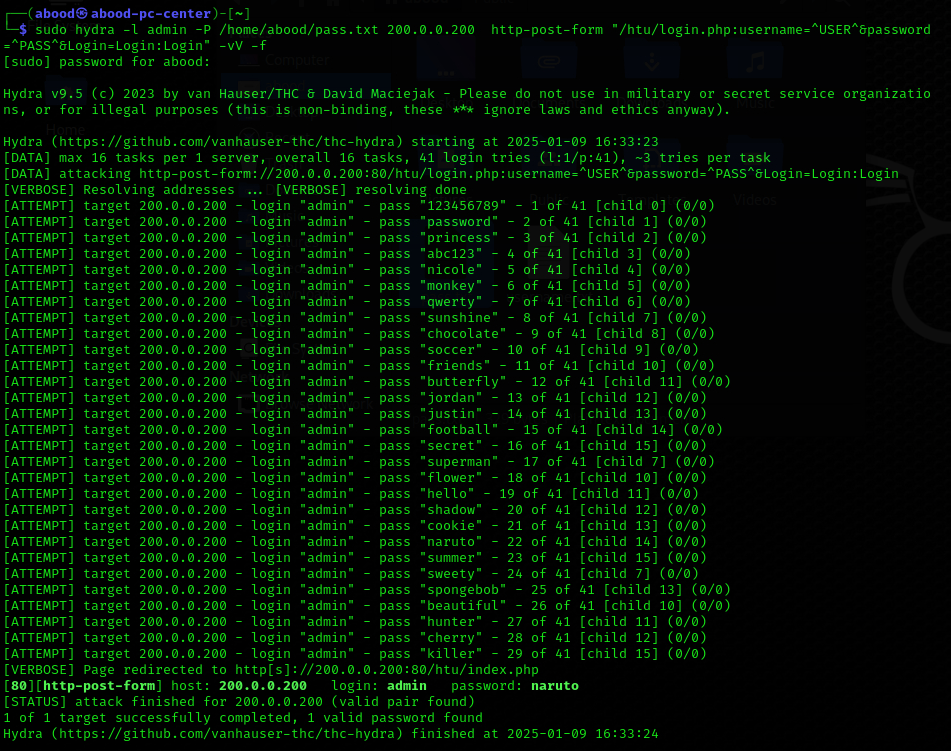


The use of FTP for file transfer poses a serious security risk, as plaintext credentials and data transmissions are susceptible to interception and unauthorized access. This can lead to compromised accounts, data leakage, and the potential for further exploitation of the affected systems.

**Recommendation**

To mitigate this risk, it is recommended to DisableFTP and migrate to a secure file transfer protocol, such as **SFTP (Secure File Transfer Protocol)** – Uses SSH to provide encryption for both data and credentials.

1. Utilize the Ajloun Teleferik server to execute a password cracking attack using Hydra



During the security assessment, a brute-force attack was successfully conducted on the target system using Hydra. The attack targeted an HTTP login form and attempted various password combinations for the username "admin". As shown in the results, the attack successfully discovered a valid login credential pair:

* **Username:** admin
* **Password:** naruto

**Recommendation**

To mitigate the risk of brute-force attacks, it is strongly advised to implement the following security controls:

* **Enforce Strong Password Policies** – Require users to create strong, complex passwords with a minimum length and a combination of uppercase, lowercase, numbers, and special characters.
* **Implement Account Lockout Mechanisms** – Lock accounts temporarily after a defined number of failed login attempts to prevent automated brute-force attacks.
* **Use CAPTCHA** – Add CAPTCHA to login forms to prevent automated tools from repeatedly attempting login attempts.
* **Enable Multi-Factor Authentication (MFA)** – Require an additional verification factor, such as a one-time password (OTP), to enhance security.

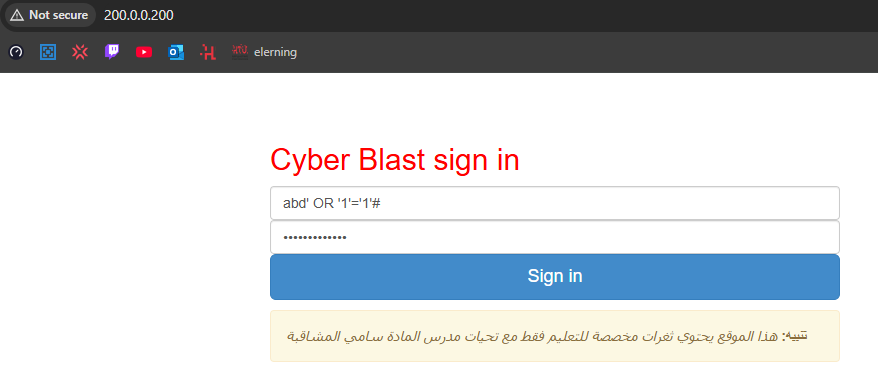
By implementing these measures, the organization can significantly reduce the likelihood of unauthorized access via brute-force attacks.

1. Utilize the Ajloun Teleferik web server to exploit the following vulnerabilities:
   1. Sql injection

As part of the assessment, we attempted an SQL injection attack on the website hosted at IP address **200.0.0.200**. The goal of this attack was to bypass authentication by exploiting improperly sanitized user inputs in the login form.

* abd’ OR ‘1’=’1’#
* pass‘OR’1’=’1

and the results are:



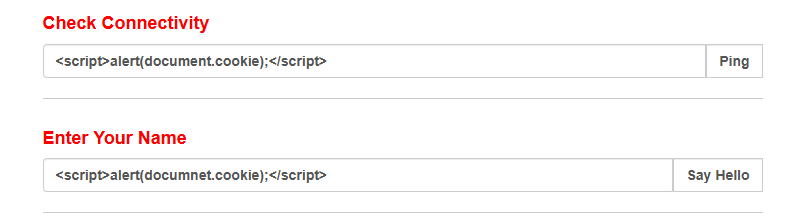


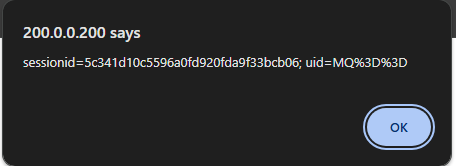
For mitigation you can find them in the common web vulnerabilities at the top of the report

* 1. XSS

The objective of this test is to determine whether the web application is vulnerable to **Cross-Site Scripting (XSS)** attacks. XSS vulnerabilities allow attackers to inject malicious scripts into web pages, which can execute in the context of other users' browsers. If successful, such an attack could be used to steal sensitive information, such as session cookies or tokens, or manipulate web content.

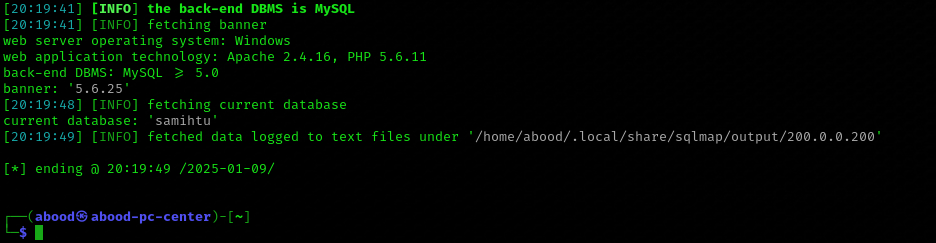
I tested three inout fields by injecting them with this command:  
<script>alert(document.cookie);</script>



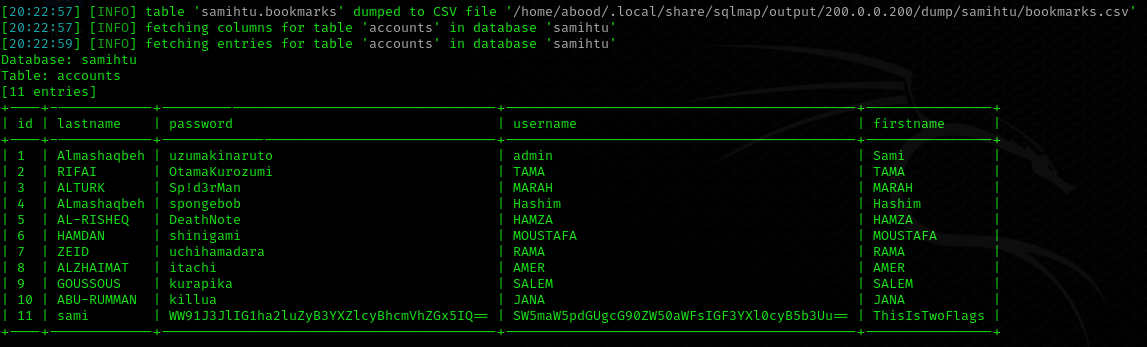
Out of the three input fields tested, two were found to be vulnerable to XSS attacks. Upon entering the payload, the script executed successfully, revealing the session cookie of the currently logged-in user through an alert box.  


As we can see this reveals the user session cookie through an alert box we can find mitigation in the common web vulnerabilities at the top of the report

* 1. Another SQL vulnerabilities using SQLmap
* Identifying the Vulnerability  
  Using SQLMAP, we tested the input fields on the Welcome Page for potential SQL injection vulnerabilities. SQLMAP successfully identified that the application was vulnerable and allowed further exploitation.
* Fetching Database Details  
  After confirming the vulnerability, we used SQLMAP to extract detailed information about the database. The following key details were retrieved (as shown in the picture under):
* **Database Management System (DBMS):** MySQL 5.6.25
* **Web Server:** Apache 2.4.16 running on **Windows**
* **Web Application Technology:** PHP 5.6.11
* **Current Database Name:** samihtu



3. Dumping the Database  
SQLMAP was used to dump the contents of the database **"samihtu"**, specifically the table **"accounts"**, which contained sensitive user information.

  
as we can see here we have many credentials that we were able to extract using sqlmap and there are 2 flags in them which isn’t part of our assignment but suspect its part of an older assignment

Now after using those commands:

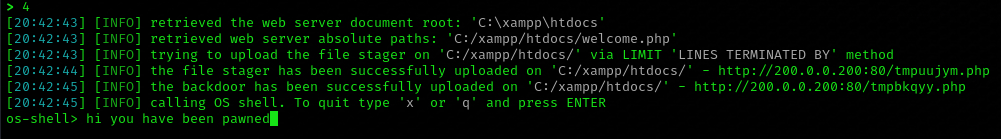
sqlmap -b -u "http://200.0.0.200/welcome.php?search=%" --cookie="sessionid=5c341d10c5596a0fd920fda9f33bcb06; uid=MQ%3D%3D;" --current-db

sqlmap -b -u "http://200.0.0.200/welcome.php?search=%" --cookie="5c341d10c5596a0fd920fda9f33bcb06; uid=MQ%3D%3D" --dump -D samihtu

to extract the previous information

lets try to access the os shell with this command

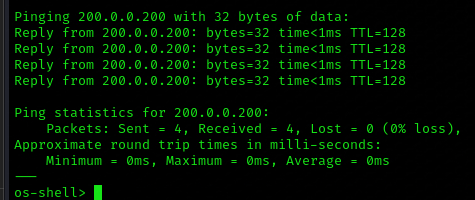
sqlmap -b -u "http://200.0.0.200/welcome.php?search=%" --cookie="sessionid=5c341d10c5596a0fd920fda9f33bcb06; uid=MQ%3D%3D;" --os-shell



And as you can see in the picture the os shell is compromised

* 1. Command injection vulnerability

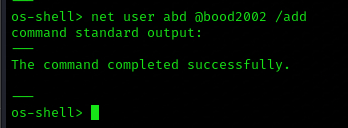
In this step we already are in the shell lets see if we can execute command



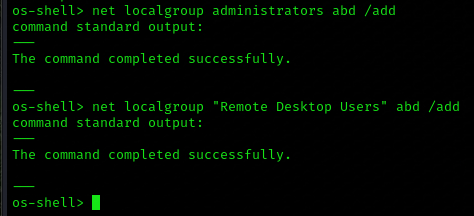
If that worked there shouldn’t be any problem executing this

net user abd @bood2002 /add

as expected, the user was added successfully



Now we have to give this user admin privileges



Now the user we added “abd” should have admin privileges

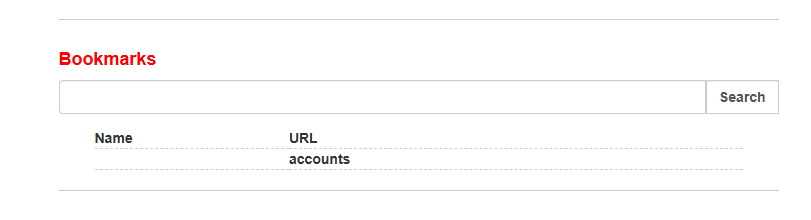
* 1. Using manual SQL to find all user accounts and passwords

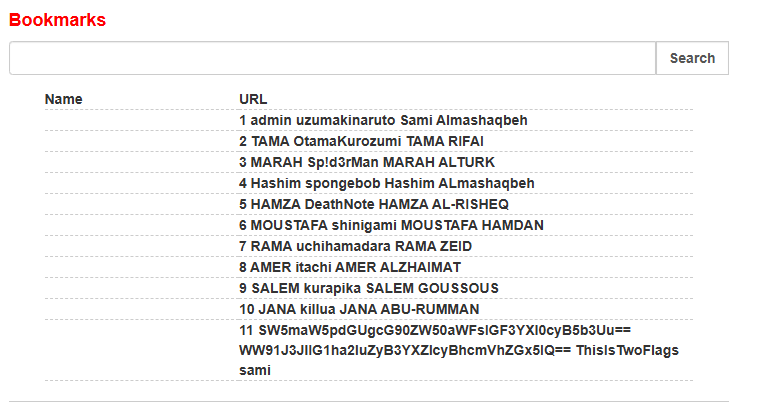
We attempted SQL injection by manually entering payloads into the search input field on the Welcome Page. The input field was found to be vulnerable, allowing us to execute SQL queries directly on the backend database.

%' and 1=0 union select null, table\_name from information\_schema.tables where table\_name like '%accounts%'#abd

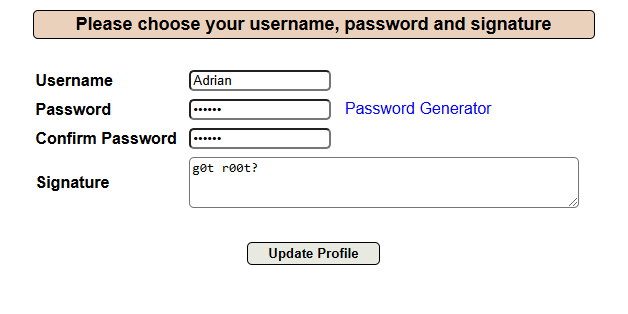
%' and 1=0 union select null, concat(id,0x0a,username,0x0a,password,0x0a,firstname,0x0a,lastname) from accounts #

The payloads successfully worked and those were the results:

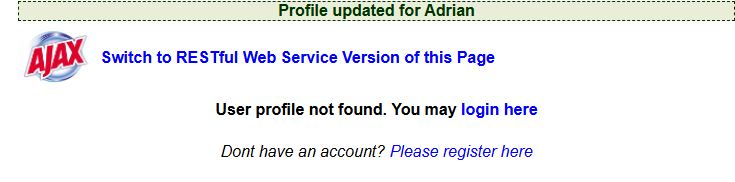




1. Utilizing the blast server to exploit top 10 vulnerabilities
2. In this task we are required to change a user’s password lets explore what we can do after loging in by doing a simple SQL injection we can see that if we want to change the user password we can input any user we want:



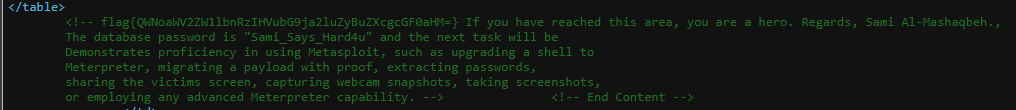
So we enter the username we want to change ‘adrian’ and we put whatever password we want



We mitigate this by passing the credintials in the login token and take the username form there

1. Search for sensitive data to discover the database password and sami’s comments

In this page we can easily find some comments left in the code written in them the database password “Sami\_says\_Hard4u”

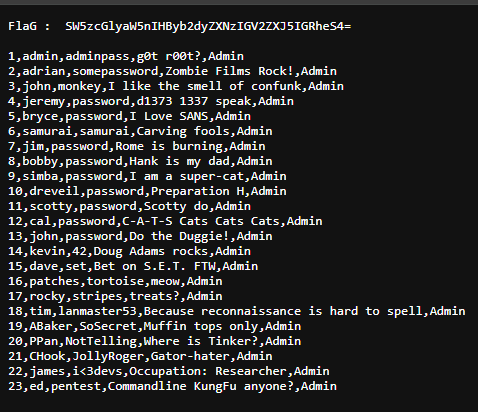


1. Find security misconfiguration find passwords directory and access it

In this task I used the tool dirb, I used it in CTFs before and comfortable with it after running it on the website it raveled many directory and one of them was:  

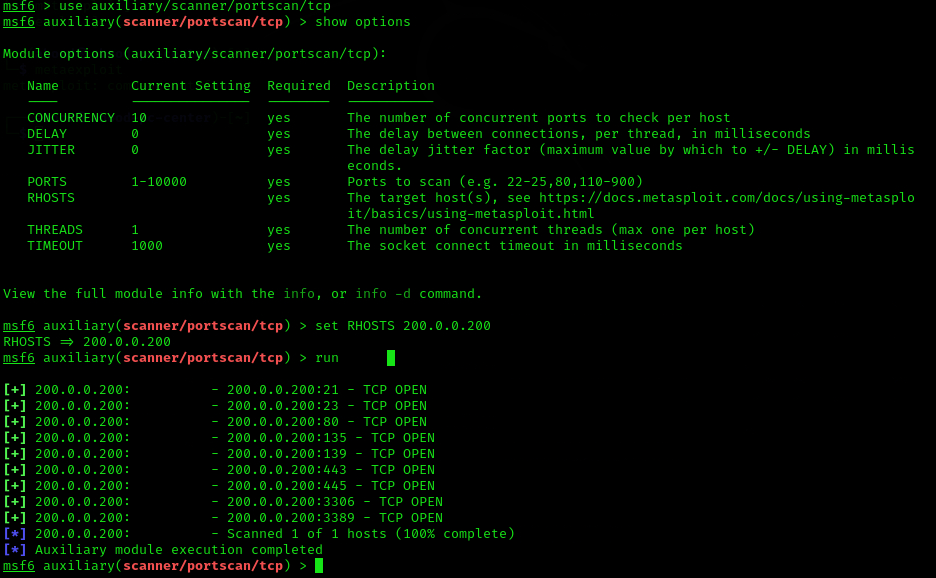

After that we headed to it to see whats inside





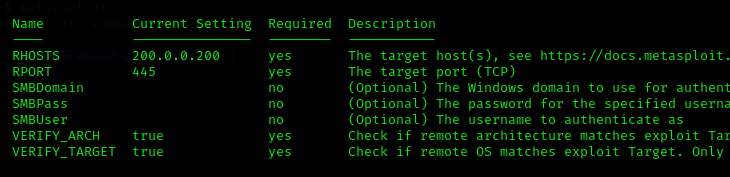
And that is the list of passwords required

1. Use ***The Metasploit framework*** to perform the following tasks on the Ajloun Teleferik server
2. Identify the vulnerable services on the server using Metasploit auxiliary modules.



I used the Metasploit framework's TCP port scanner module to scan the target 200.0.0.200. The scan identified multiple open TCP ports, including 21, 22, 25, 80, 443, and more.

1. Exploit well-known vulnerabilities that you find, such as EternalBlue or the VSFTPD backdoor, using Metasploit modules.

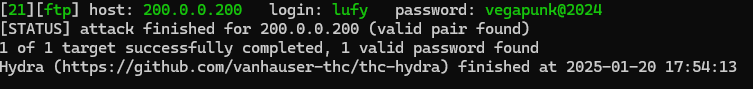




I exploited the SMB vulnerability MS17-010 (EternalBlue**)** on the target 200.0.0.200. The exploit successfully created a Meterpreter session, granting remote access to the target system.

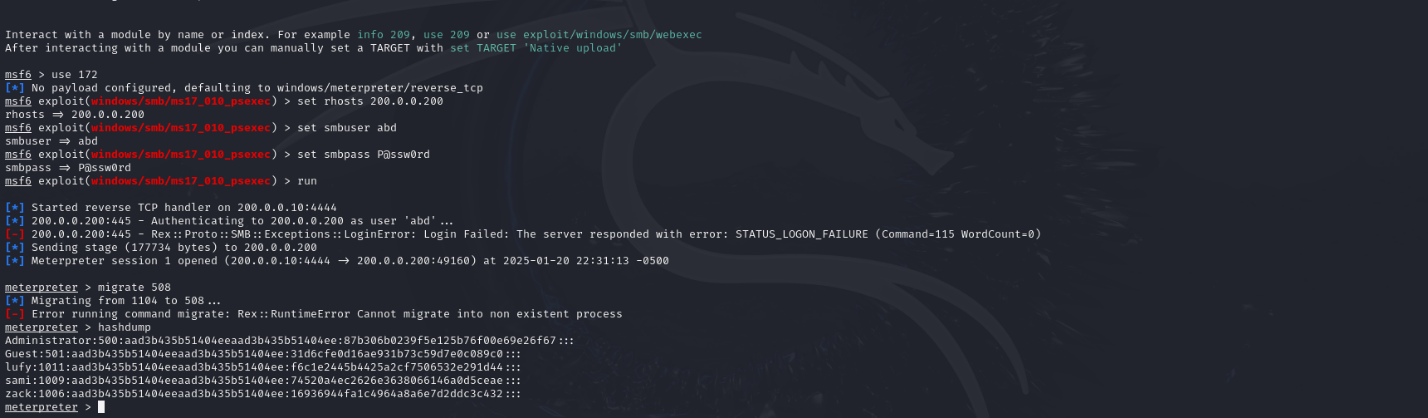
1. Use the auxiliary module or Hydra to brute force the FTP service and find the login password using the provided username and password list.





I used Hydra to perform a brute-force attack on the FTP service of the target 200.0.0.200 using username and password lists provided. The attack successfully identified valid credentials: **username:** lufy, **password:** vegapunkQ2024.

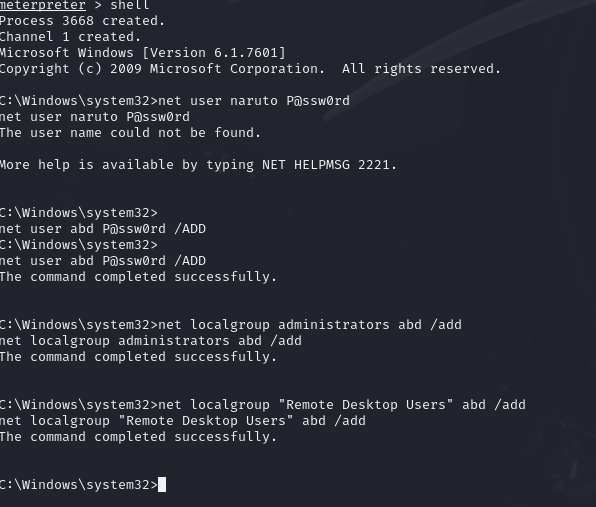
1. Migrate your payload with proof, extracting passwords (don’t forget to de-hash the passwords), sharing the victim's screen, and taking screenshots remotely from the victim machine

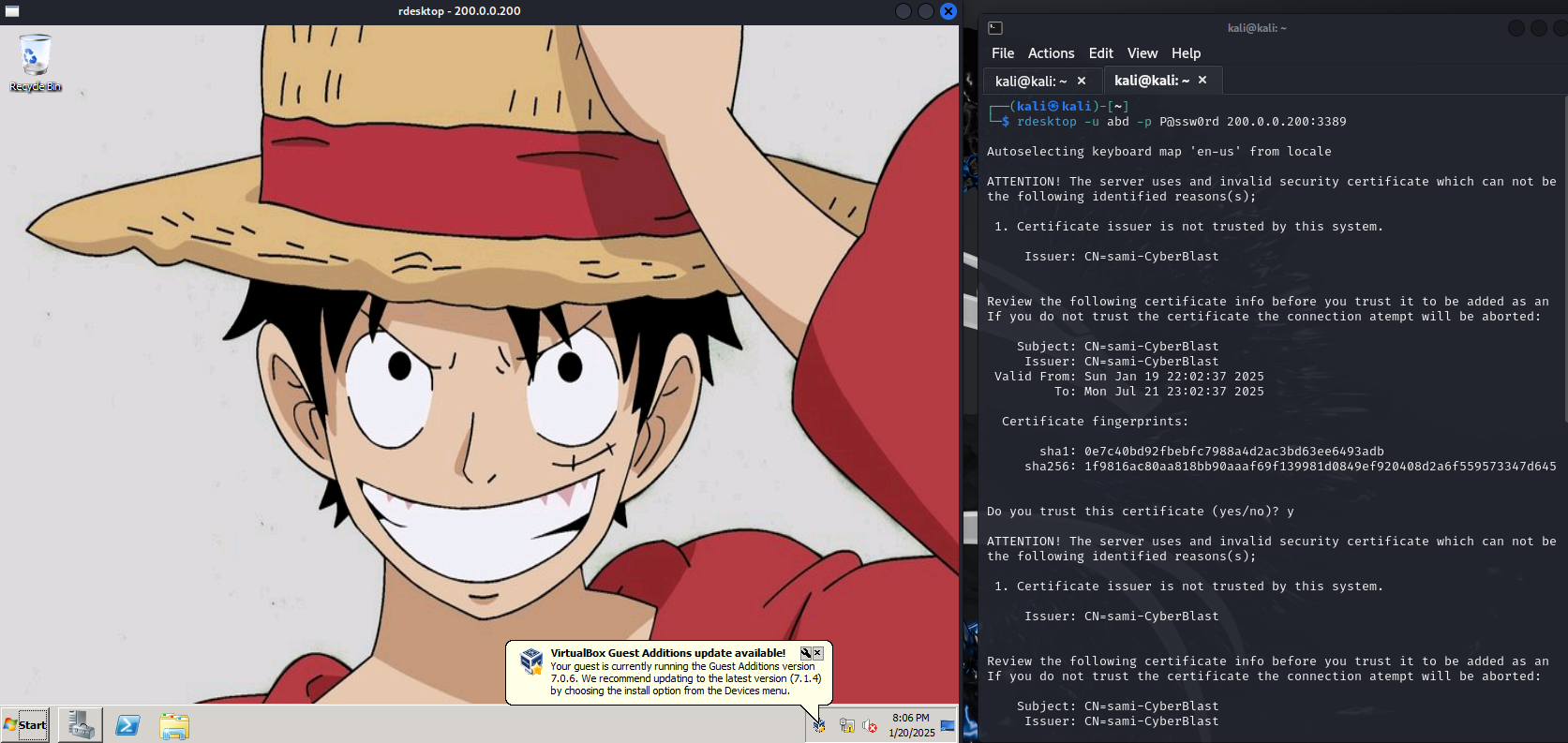




I used the windows/smb/ms17\_010\_psexec module to exploit the MS17-010 vulnerability on the target 200.0.0.200. After successfully opening a Meterpreter session, I migrated the payload to process ID 508

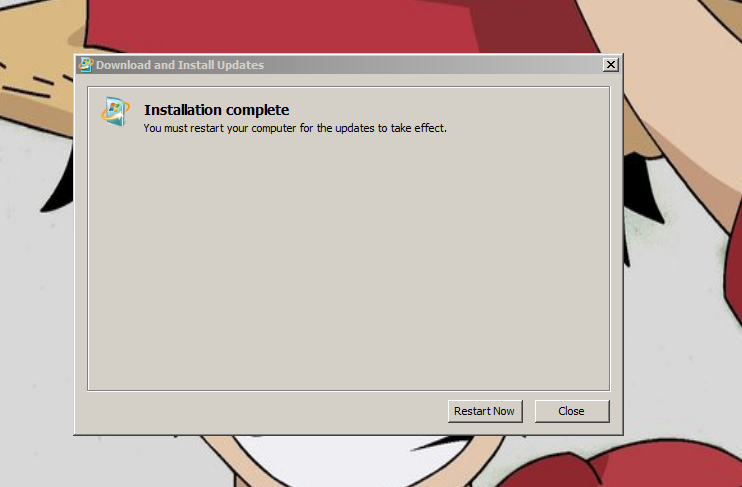
1. Create a username (you may use your name) and add the user to the local 'Administrator' and 'Remote Desktop Users' groups. Connect to the target machine using the created username and password through Remote Desktop Connection.

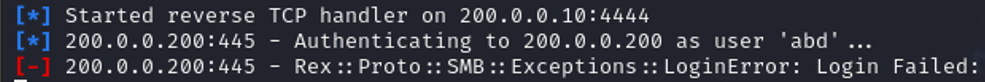




I used the Meterpreter shell to create a new user abd with the password P0ssw0rd, added it to the Administrators and Remote Desktop Users groups, and then used the rdesktop tool to remotely connect to the victim's machine. Once connected, I verified access by sharing the victim's desktop and took screenshots of the system as proof of successful compromise.

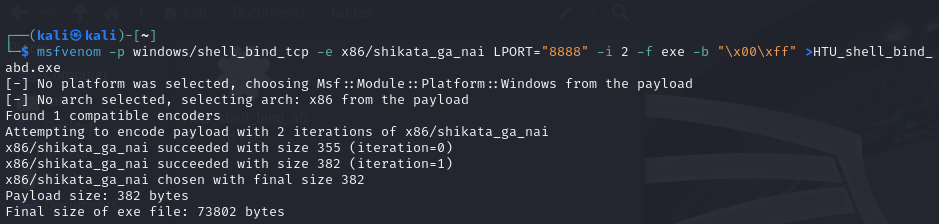
1. Utilize the patch file located on the targeted server desktop to address this vulnerability. Subsequently, employ Metasploit auxiliary or attempt to exploit the vulnerability again. Provide evidence demonstrating that you are unable to exploit it after applying the patch.

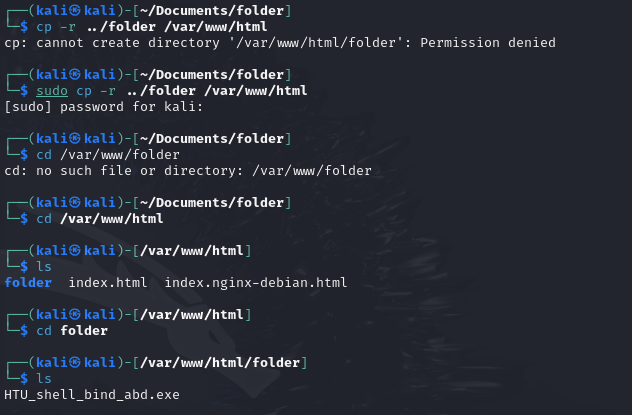




I patched the the system using the file and tried to redo the attack and it failed

1. Use the auxiliary module or Hydra to brute force the FTP service and find the login password using the provided username and password list.

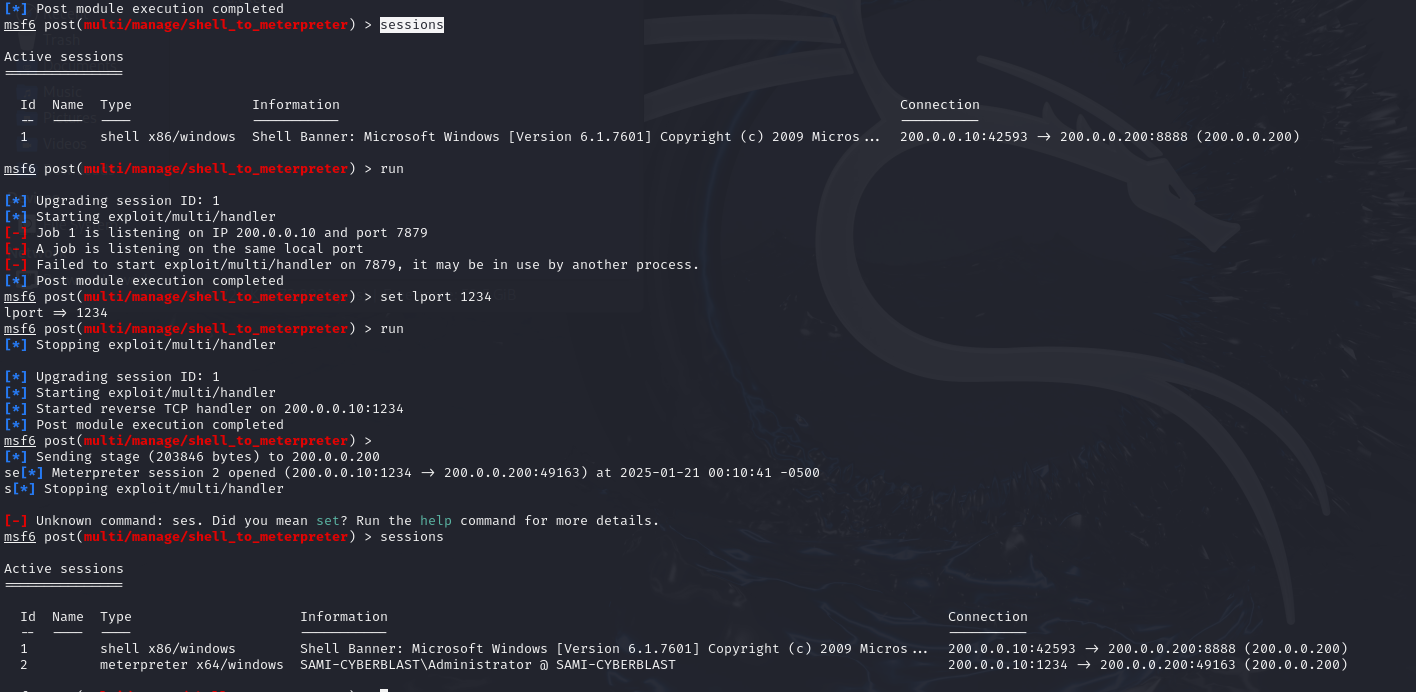


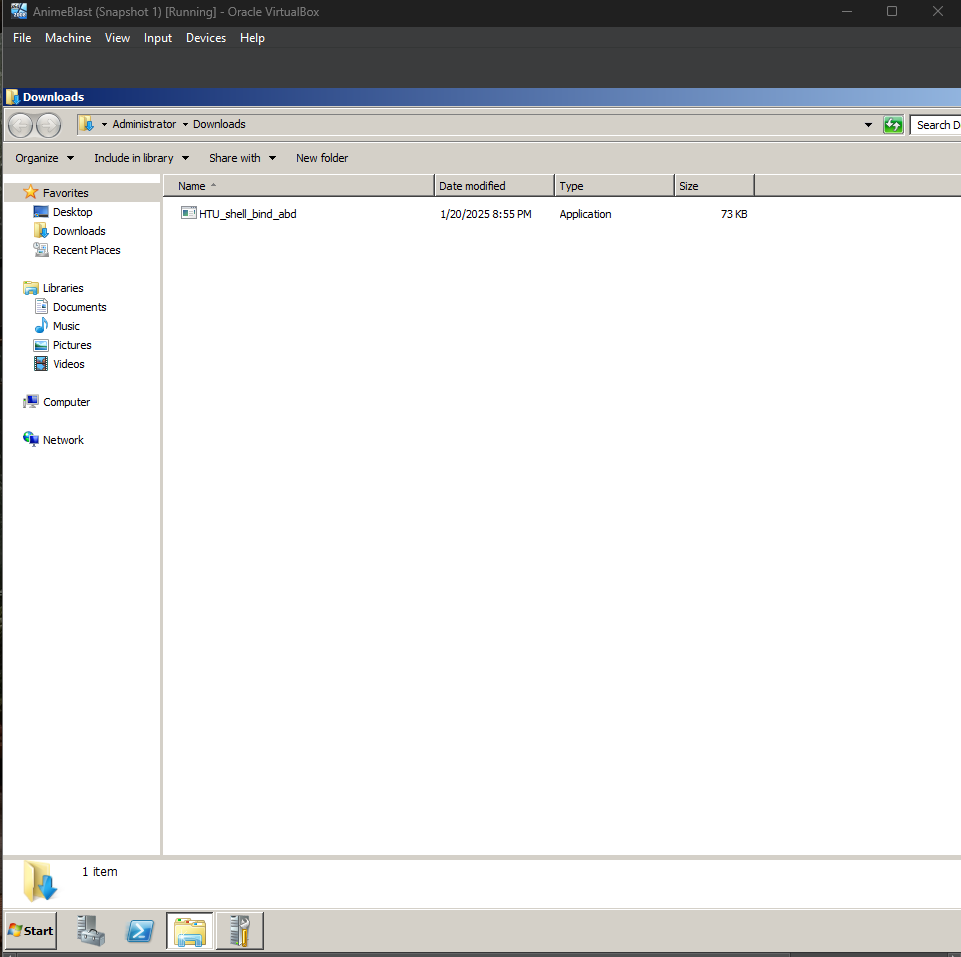








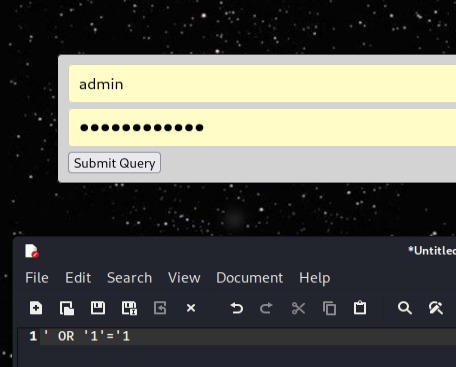


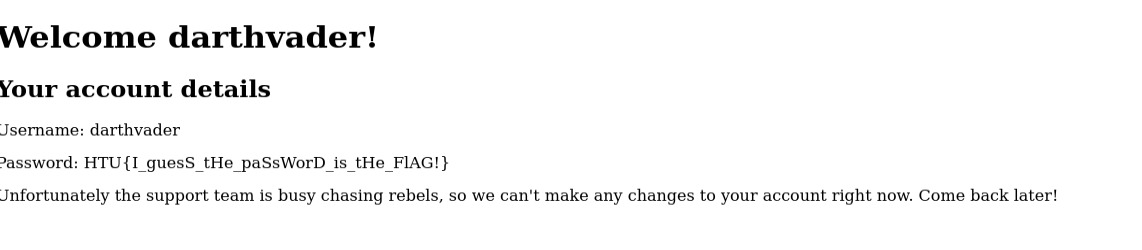


I created a custom payload using msfvenom with a bind shell configuration encoded with shikata\_ga\_nai. The payload was saved as an executable file (HTU\_shell\_bind\_abd.exe) and hosted on a web server using Apache. After transferring and executing the payload on the target machine, I established a bind shell connection on port 8888. Using Metasploit's multi/handler, I upgraded the bind shell to a Meterpreter session for enhanced functionality, verified the session stability, and captured proof of execution on the victim machine through screenshots of the running payload.

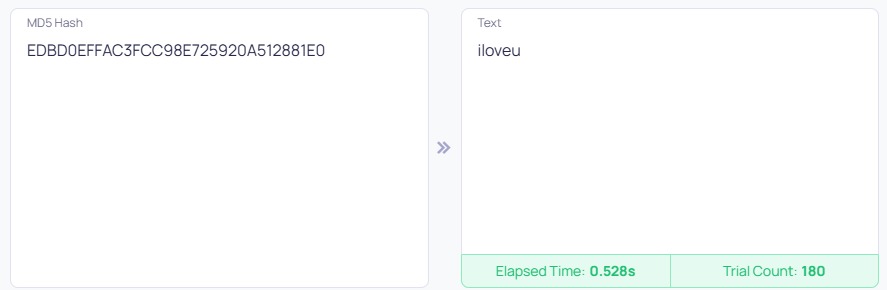
CTF flags:

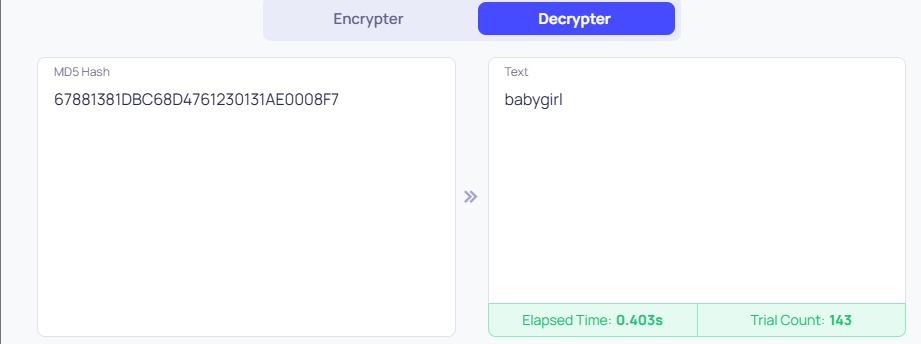
* 1. Challenge: boardeeeeng:



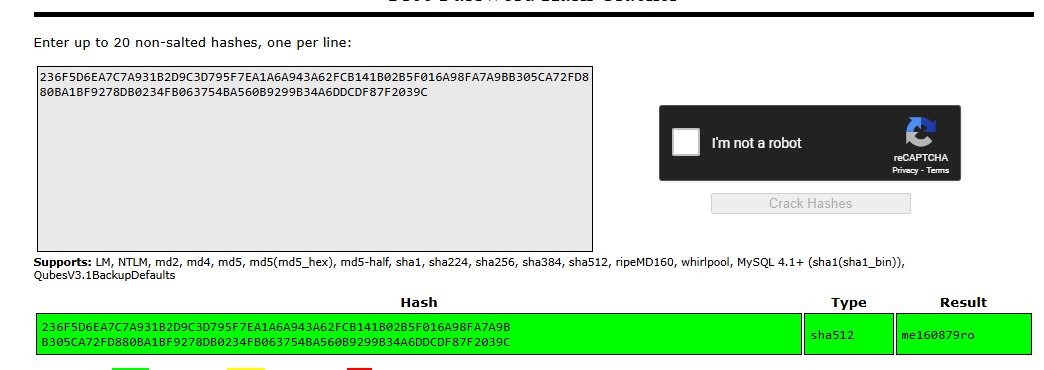


* 1. Md5

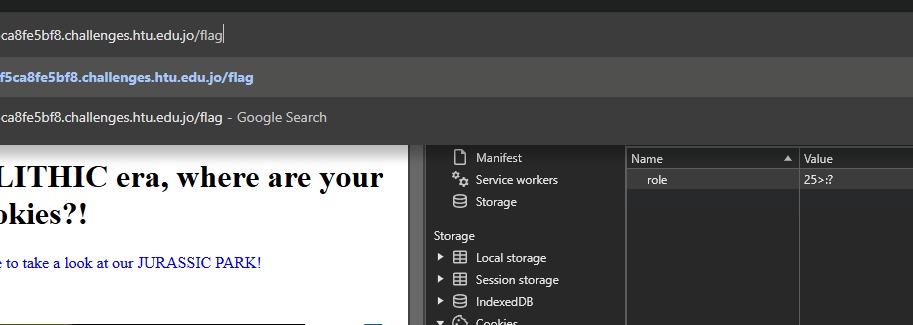




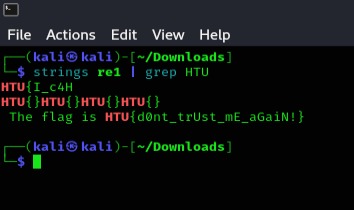
* 1. Sha512:



* 1. Client cookie:

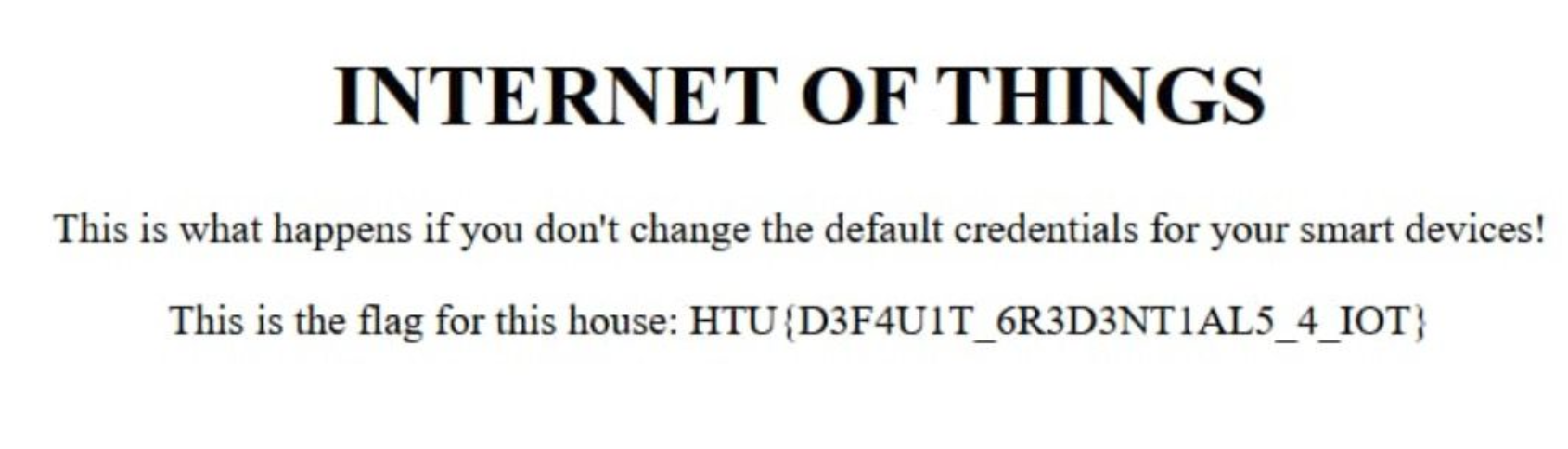


* 1. Re1

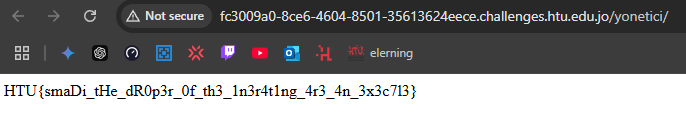


* 1. ThingOfInternet

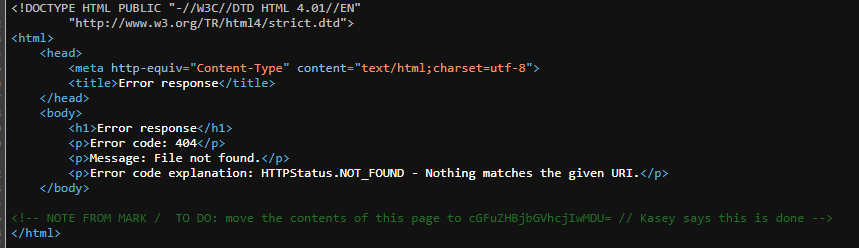
Here the password was the default of the fridge in the background



* 1. Oh my blog

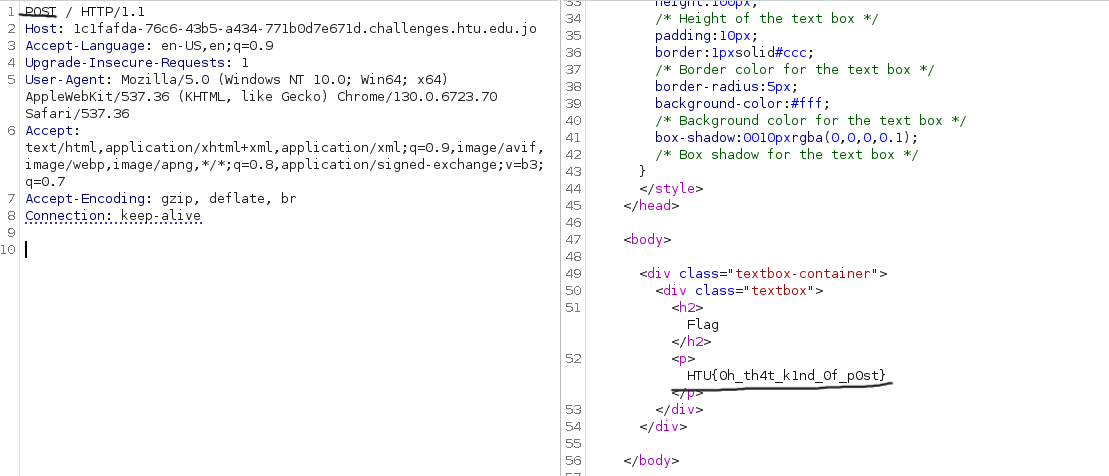


* 1. Shangirla

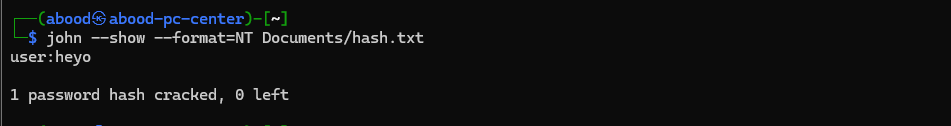
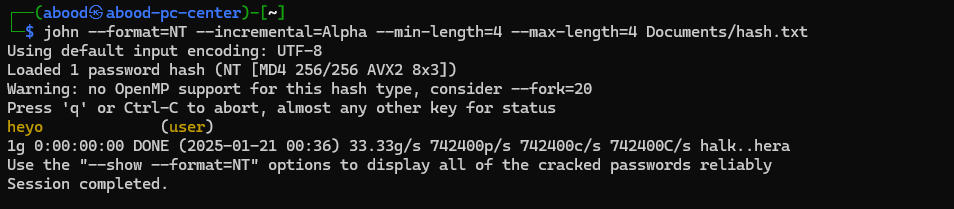


* 1. smadiPost

using burpsuite change the “GET” to “POST”



* 1. NTLM



**Reference list**

1. Ethical hacking slides