CVE Hunting & Exploit Development Quick Reference

Lab Setup Essentials

Debian Target VM Setup

bash

```
# Download Debian 10.13 ISO
# https://cdimage.debian.org/cdimage/archive/10.13.0/amd64/iso-dvd/

# VM Configuration
# OS Type: Linux → Debian (64-bit)
# RAM: 2 GB minimum
# Network: Internal Network (intnet)
# Storage: 20 GB (dynamic)
```

Network Configuration

bash

```
# Check IP address
ip a

# Ping test between machines
ping 192.168.2.191
```

Basic System Updates

bash

```
# Update package index only (keep vulnerable)
sudo apt update

# Install essential tools
sudo apt install build-essential git gcc vim curl -y

# ⚠️ DO NOT run: sudo apt upgrade
# Keep system vulnerable for testing
```

SSH Service Setup & Enumeration

Target SSH Configuration

bash

```
# Install and enable SSH
sudo apt install openssh-server -y
sudo systemctl enable ssh && sudo systemctl start ssh

# Create test users
sudo adduser testuser
sudo adduser student
sudo adduser rootme
```

SSH User Enumeration

bash

```
# Create username wordlist
echo -e "root\nadmin\nstudent\ntestuser\nguest\ndeveloper" \
> userlist.txt

# Enumerate valid users with Hydra
hydra -L userlist.txt -p invalidpassword -t 1 -V \
192.168.2.191 ssh

# Alternative single user test
hydra -l testuser -p wrongpass -t 1 -V 192.168.2.191 ssh
```

CVE Research & Analysis

CVE Information Gathering

bash

```
# Search CVE databases
# - https://cve.mitre.org/
# - https://nvd.nist.gov/
# - https://www.exploit-db.com/

# Check software versions
ssh -V  # Check SSH version
cat /etc/os-release  # Check OS version

# Service version enumeration
nmap -sV -p22 192.168.2.191
```

Exploit Development Workflow

bash

```
# 1. Find CVE and PoC code
# 2. Analyze vulnerability details
# 3. Set up vulnerable environment
# 4. Test and modify PoC
# 5. Develop reliable exploit

# Common PoC sources:
# - GitHub repositories
# - ExploitDB
# - Security advisories
# - Research papers
```

Network Scanning & Service Discovery

Basic Network Enumeration

bash

```
# Quick port scan
nmap -sS -O 192.168.2.191

# Service version detection
nmap -sV -p- 192.168.2.191

# OS fingerprinting
nmap -O 192.168.2.191

# Script scanning for vulnerabilities
nmap --script vuln 192.168.2.191
```

SSH Specific Scanning

```bash
# SSH version detection
nc 192.168.2.191 22

# SSH configuration analysis
nmap --script ssh-auth-methods 192.168.2.191
nmap --script ssh-hostkey 192.168.2.191

# Check for SSH vulnerabilities
nmap --script ssh* 192.168.2.191
```

Exploit Development Tools

Compilation & Building

```bash
```

```
# Compile C exploits
gcc -o exploit exploit.c

# With specific flags
gcc -o exploit exploit.c -lpthread

# Python exploit execution
python3 exploit.py target_ip

# Make executable
chmod +x exploit.py
./exploit.py
```

Debugging & Analysis

bash

```
# Check running processes
ps aux | grep ssh

# Monitor system logs
sudo tail -f /var/log/auth.log

# Check network connections
netstat -tulpn | grep :22

# System resource monitoring
htop
```

User Enumeration Techniques

Manual SSH Testing

bash

```
# Test single username
ssh testuser@192.168.2.191

# Check login timing differences
time ssh nonexistent@192.168.2.191
time ssh validuser@192.168.2.191
```

Hydra User Enumeration

bash

```
# Basic user enumeration
hydra -L userlist.txt -p fakepass 192.168.2.191 ssh

# Verbose output
hydra -L userlist.txt -p fakepass -V 192.168.2.191 ssh

# Single thread (stealth)
hydra -L userlist.txt -p fakepass -t 1 192.168.2.191 ssh
```

Vulnerability Assessment

SSH Configuration Checks

bash

```
# Check SSH config file
sudo cat /etc/ssh/sshd_config

# Look for misconfigurations:
# - PermitRootLogin yes
# - PasswordAuthentication yes
# - PermitEmptyPasswords yes
# - PubkeyAuthentication no
```

System Information Gathering

bash

```
# Kernel version (for privilege escalation)
uname -a

# Check installed packages
dpkg -l | grep ssh

# Check system users
cat /etc/passwd

# Check sudo permissions
sudo -l
```

Common CVE Categories

SSH Vulnerabilities

- **User enumeration** (CVE-2018-15473)
- **Authentication bypass**
- **Buffer overflow vulnerabilities**
- **Configuration weaknesses**

Research Priority

1. **Information disclosure** (easy to exploit)
2. **Authentication bypass** (high impact)
3. **Remote code execution** (critical)
4. **Privilege escalation** (post-exploitation)

Exploit Modification Tips

Common PoC Issues

- **Hardcoded IP addresses** - Replace with target IP
- **Python version conflicts** - Check Python 2 vs 3
- **Missing dependencies** - Install required libraries
- **Network timeouts** - Adjust timing parameters
- **Payload encoding** - Fix character encoding issues

Making PoCs Reliable

```bash
# Add error handling
try:
    exploit_function()
except Exception as e:
    print(f"Exploit failed: {e}")

# Add timing delays
import time
time.sleep(1)

# Test connectivity first
import socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
result = sock.connect_ex((target_ip, target_port))
if result == 0:
    print("Port is open")
```

Best Practices

Ethical Guidelines

- Only test on systems you own or have permission to test
- Use isolated lab environments
- Document all findings for defensive purposes
- Follow responsible disclosure for new vulnerabilities

Lab Management

- Take VM snapshots before testing
- Keep vulnerable systems isolated from networks
- Document successful exploitation steps
- Save working exploits with clear comments

Research Methodology

1. **Identify target software and versions**

2. **Search for known CVEs and PoCs**
3. **Set up vulnerable test environment**
4. **Analyze and modify exploit code**
5. **Test reliability and document results**