Practical Lab Using Nmap and Scapy

1. Introduction

During our recent cybersecurity practical sessions, we explored two fundamental network security tools: **Nmap** for network discovery and vulnerability assessment, and **Scapy** for packet manipulation and analysis. This documentation presents a structured account of our hands-on experience, demonstrating the practical application of these tools in real-world cybersecurity scenarios.

The exercises were conducted within a controlled and authorized lab environment to ensure ethical and legal compliance with security best practices.

2. Lab Environment

2.1 Tools Utilized

- **Nmap (Network Mapper) v7.92** – Network discovery and scanning
- **Scapy** – Python-based packet crafting and analysis tool
- **Wireshark** – Graphical packet analysis tool
- **Kali Linux VM** – Cisco Ethical Hacking lab environment
- **tcpdump** – Command-line packet capture utility

2.2 Network Configuration

- **Target Subnet:** 10.6.6.0/24
- **Primary Target Host:** 10.6.6.23

3. Part One: Network Reconnaissance with Nmap

3.1 Host Discovery

```
nmap -sn 10.6.6.0/24
```

Objective:
To perform a ping sweep across the subnet and identify live hosts without conducting port scans. This initial reconnaissance phase helps establish a network map before deeper analysis.

3.2 Operating System Detection

```
sudo nmap -O 10.6.6.23
```

Objective:
To fingerprint the target’s operating system by analyzing TCP/IP stack behaviors. OS identification supports informed decision-making when selecting security controls or attack vectors.

3.3 Service and Version Detection

```
nmap -p21 -sV -A -T4 10.6.6.23
```

Command Breakdown:

- -p21: Scan FTP port only
- -sV: Detect service versions
- -A: Enable OS detection, script scanning, traceroute
- -T4: Aggressive timing template

Objective:
To identify running services and versions in order to detect potential vulnerabilities associated with outdated software.

3.4 SMB Service Enumeration

```
nmap -p139,445 10.6.6.23
nmap --script smb-enum-shares.nse -p445 10.6.6.23
```

Objective:
To identify open SMB ports and enumerate available network shares. SMB services are frequently targeted during penetration testing due to common misconfigurations.

3.5 SMB Client Interaction

```
smbclient //10.6.6.23/print$ -N
```

Objective:
To connect anonymously to a shared resource and test access permissions. The session was closed using the exit command.

3.6 Local Network Verification

```
ifconfig
ip route
cat /etc/resolv.conf
```

Objective:
To validate local network configuration prior to active scanning activities.

4. Packet Capture and Analysis

4.1 Traffic Capture Using tcpdump

```
sudo tcpdump -i eth0 -s 0 -w ladies.pcap
# Ctrl + C to stop capture
ls ladies.pcap
```

4.2 Packet Analysis with Wireshark

```
wireshark ladies.pcap
```

Objective:
To capture live network traffic using tcpdump and analyze it using Wireshark’s graphical interface. This workflow demonstrates the efficiency of command-line capture combined with in-depth GUI-based protocol analysis.

5. Part Two: Packet Manipulation with Scapy

5.1 Basic Packet Sniffing

```
sudo su
scapy
sniff()
```

Procedure:

1. Start sniffing packets within Scapy
2. Generate traffic using: ping google.com
3. Stop capture using Ctrl + C

```
paro = _
paro.summary()
```

Outcome:
Displays summarized packet information including source/destination IP addresses and protocols.

5.2 Interface-Specific Sniffing

```
sniff(iface="br-internal")
```

Traffic Generation:

- Ping sweep: ping 10.6.6.1/24
- Web access: Browse to 10.6.6.23

```
paro2 = _
paro2.summary()
```

5.3 Filtered Packet Capture

```
sniff(iface="br-internal", filter="icmp", count=5)
```

Objective:
To capture only ICMP packets, demonstrating Scapy’s filtering capability for targeted analysis.

5.4 Packet Inspection

```
paro3 = _
paro3.summary()
paro3[3]
```

Objective:
To inspect individual packets and analyze protocol fields in detail.

6. Key Learnings

- Nmap provides comprehensive network visibility beyond simple port scanning
- Structured scanning methodology improves reconnaissance efficiency
- Scapy enables deep protocol-level packet manipulation
- Combining tcpdump, Wireshark, and Scapy provides a powerful analysis workflow
- Understanding packet structures enhances security investigation capabilities
- Ethical authorization is essential before performing any network scanning

7. Challenges Encountered

- Requirement for administrative privileges on scanning tools
- Interpreting complex scan results accurately
- Analyzing raw packet data without visualization
- Memorizing extensive command syntax
- Switching between CLI and GUI tools

8. Tool Comparison

Wireshark

Strengths:

- Intuitive graphical interface
- Advanced display filtering
- Extensive protocol support
- Excellent offline file analysis

Scapy

Strengths:

- Full control over packet construction
- Automation through Python scripting
- Real-time interactive testing
- Strong educational value

---

9. Practical Workflow

1. Capture traffic: tcpdump -w capture.pcap
2. Visual analysis: Open in Wireshark
3. Packet manipulation: Use Scapy
4. Validation: Re-capture traffic for verification

---

10. Real-World Applications

For Cybersecurity Professionals

- Network asset discovery
- Vulnerability assessments
- Incident response analysis
- Compliance auditing
- Security research and testing
- Digital forensics

For Organizations

- Attack surface reduction
- Regulatory compliance
- Threat modeling
- Security awareness demonstrations

11. Conclusion

This practical exercise provided invaluable hands-on experience with industry-standard security tools. Nmap enabled structured network reconnaissance, while Scapy delivered deep insight into packet-level communication. The combination of tcpdump and Wireshark further strengthened analytical capabilities. Together, these tools form a powerful toolkit for modern cybersecurity professionals, reinforcing both technical proficiency and ethical responsibility.