SEED Labs Report: The Mitnick Attack

Name: Muhammad Ammar Shahid
Course: Net Sec
Roll Number: 23i-2125
Class: Cy-A
Date: 15/9/2025

Table of Contents

1. Introduction

The Mitnick Attack is one of the most famous early cyberattacks, exploiting weaknesses in TCP/IP and trust-based authentication. Kevin Mitnick silenced a trusted server using SYN flooding, then impersonated it to establish fake connections with an X-Terminal, injecting commands without authentication. This lab recreates the attack in a containerized environment to demonstrate TCP spoofing, ARP manipulation, and .rhosts vulnerabilities.

Objectives:

- Understand how SYN flooding and ARP caching can be used in spoofing.
- Use Scapy to spoof TCP handshakes.
- Recreate rsh-based remote command injection.
- Plant a backdoor for persistent unauthorized access.

2. Lab Setup

- **VM:** SEED Ubuntu 20.04 in VirtualBox
- **Tools:** Docker, docker-compose, Scapy, tcpdump/Wireshark
- **Network:**
    - Attacker: 10.9.0.1
    - X-Terminal: 10.9.0.5
    - Trusted Server: 10.9.0.6

3. Task 1 – Simulated SYN Flooding

Goal: Silence Trusted Server while ensuring its MAC stays cached in X-Terminal.

Steps:

1. Added static ARP entry:
2. arp -s 10.9.0.6 <MAC>
3. Stopped Trusted Server:
4. docker stop trusted-server-10.9.0.6

Observation: X-Terminal continued using cached MAC, enabling spoofing.

4. Task 2 – Sniff & Spoof TCP Connections

4.1 First Connection

- Spoofed SYN from 10.9.0.6:1023 → 10.9.0.5:514.
- Sniffed SYN+ACK, replied with spoofed ACK.
- Injected rsh data to run:
- touch /tmp/xyz

Key Code Snippet:

data = b"9090\x00seed\x00seed\x00touch /tmp/xyz\x00"

4.2 Second Connection

- X-Terminal attempted second connection to port 9090.
- spoof_err.py replied with SYN+ACK, completing handshake.
- rshd then executed the injected command.

Observation: Without this second connection, commands were ignored. With it, /tmp/xyz was created successfully.

5. Task 3 – Planting a Backdoor

Goal: Persistent access without repeating spoofing attack.

Injected Command:

echo + + > /home/seed/.rhosts; chmod 644 /home/seed/.rhosts

Steps:

1. Modified spoof1.py payload with the above command.
2. Ran spoof1 and spoof_err again.
3. Verified .rhosts contained + +.
4. Installed rsh client on attacker:
5. apt-get install -y rsh-redone-client
6. Logged in from attacker without password:
7. rsh 10.9.0.5 -l seed

6. Observations

- ARP caching was critical: without static entry, spoofing failed.
- rsh requires *two TCP connections*; otherwise rshd halts.
- Modern OSes randomize TCP sequence numbers, making blind spoofing infeasible.
- Planting + + in .rhosts is a major backdoor risk, showing why rsh was abandoned for ssh.

7. Conclusion

Through this lab:

1. Simulated SYN flooding to silence a trusted server.
2. Used Scapy to spoof TCP connections and inject commands.
3. Proved successful attack by creating /tmp/xyz.
4. Installed a persistent backdoor via .rhosts.

Key Lesson: Trust-based authentication and predictable sequence numbers are insecure. This lab demonstrated why legacy protocols like rsh are unsafe, and why secure alternatives (SSH) are mandatory today.