

HANDS-ON LAB: BUILD A MINI 'RECON PIPELINE' FOR YOUR VM NETWORK (PING SWEEP + PORT SCANS + SIMPLE AUTOMATION)

This lab walks through a practical security assessment workflow: discovering active hosts, identifying the services they expose, analysing port states such as open, closed, and filtered, comparing different scan techniques, and streamlining repeated tasks with a shell script. The exercise reinforces the objectives from Week 3 and applies the scanning principles covered in that week's lecture.

What you will produce (in a `recon_lab/` folder)

- `live_hosts.txt` (ping sweep results)
- Nmap outputs (Metasploitable, plus optional Windows 11)
- `summary_report.txt` (auto-generated)
- `recon_pipeline.sh` (automation script)
- A short findings summary and reflection (your write-up)

Part 0: Setup and safety boundaries

1) Use an isolated virtual network

- In VirtualBox or VMware, set **Kali**, **Metasploitable**, and **Windows 11** to the same:
 - **Host-only** or **Internal Network** (recommended)
- Confirm you can ping between VMs once you know the IPs.

2) Only scan your lab

- Keep **all targets inside your VM network**.
- Do not scan anything you do not own or have explicit permission to test.

Part 1: Identify your network range and discover target hosts

Step 1: Determine Kali IP address and subnet

On **Kali**, run:

```
root@osboxes:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:79:26:d7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.39.3/24 brd 192.168.39.255 scope global dynamic noprefixroute eth0
        valid_lft 374sec preferred_lft 374sec
    inet6 fe80::a00:27ff:fe79:26d7/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
root@osboxes:~# ip r
192.168.39.0/24 dev eth0 proto kernel scope link src 192.168.39.3 metric 100
```

Record:

- Your active interface IP address (example: 192.168.56.101)
- The subnet from the routing table (example: 192.168.56.0/24)

This tells you the network range you will scan.

Step 2: Discover other live hosts from Kali (do not log into the targets)

Instead of checking IP addresses directly on Metasploitable or Windows, identify them through network discovery.

Run a ping sweep from Kali across your subnet (adjust to your network range):

```
File Edit View Search Terminal Help
root@osboxes:~# sudo fping -a -g 192.168.56.1 192.168.56.254 2>/dev/null
```

Save the results using this command:

```
root@osboxes: ~  
root@osboxes:~# sudo fping -a -g 192.168.56.1 192.168.56.254 2>/dev/null | tee live_hosts.txt
```

You should see multiple responding IP addresses. One will be Kali itself, and the others should include Metasploitable and Windows.

Step 3: Identify which host is which

To determine which discovered IP belongs to which system, perform light enumeration from Kali: Example:

```
root@osboxes: ~  
root@osboxes:~# sudo nmap -O <IP_address>
```

or

```
root@osboxes: ~  
root@osboxes:~# sudo nmap -sS -sV --top-ports 50 <IP_address>
```

Look for clues:

- Many vulnerable services (FTP, Telnet, old Apache versions) typically indicate Metasploitable.
- Modern Windows services (SMB, RPC, Microsoft-specific ports) typically indicate Windows 11.
- Compare operating system detection results.

Document:

- Which IP you believe is Metasploitable and why.
- Which IP you believe is Windows 11 and why.
- What evidence from your scans supports your conclusion?

Why this matters

In a real assessment, you rarely log into systems to learn their IP addresses. You must discover and identify hosts remotely before performing deeper analysis. This exercise mirrors that process by requiring network-based identification rather than direct inspection.

Part 2: Run a ping sweep and note limitations

Option A (Kali): sweep with `fping`

Pick a small range that definitely includes your VMs (adjust to your subnet):

```
root@osboxes: ~  
root@osboxes:~# sudo fping -a -g 192.168.56.1 192.168.56.50 2>/dev/null
```

Save it:

```
root@osboxes: ~  
root@osboxes:~# sudo fping -a -g 192.168.56.1 192.168.56.50 2>/dev/null | tee live_hosts.txt
```

Step 2: test “ping sweep problems”

- Toggle Windows Firewall on and off, or ICMP echo settings (only if you know how to revert).
- Re-run the sweep and note if a host disappears even though it is still online.

Include 2-3 sentences in your write-up:

- Which hosts responded?
- Did any live host fail to respond to ICMP?
- Why can ICMP discovery miss hosts?

Part 3: Port scanning Metasploitable with multiple scan types

Use Metasploitable as the main target (example: 192.168.56.102).

Step 1: baseline scan (SYN + service detect + OS guess)

```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# sudo nmap -sS -sV -O --top-ports 200 192.168.56.102 -oN metasploitable_baseline.txt
```

Step 2: compare scan types (same target)

TCP connect scan:

```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# sudo nmap -sT --top-ports 200 192.168.56.102 -oN metasploitable_connect.txt
```

NULL scan:

```
File Edit View Search Terminal Help
root@osboxes: ~# sudo nmap -sN --top-ports 200 192.168.56.102 -oN metasploitable_null.txt
```

XMAS scan:

```
File Edit View Search Terminal Help
root@osboxes: ~# sudo nmap -sX --top-ports 200 192.168.56.102 -oN metasploitable_xmas.txt
```

Step 3: interpret open, closed, and filtered

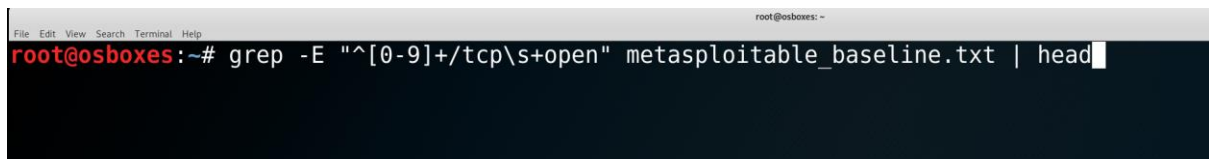
In each output file, find port states:

- **open**: a service is listening and reachable
- **closed**: host is reachable, but nothing is listening there
- **filtered**: something is blocking or interfering (firewall or filtering device)

Mini-task to record in your notes:

- 5 example **open** ports and the service names
- 2 **closed** ports
- Any **filtered** ports (if present)

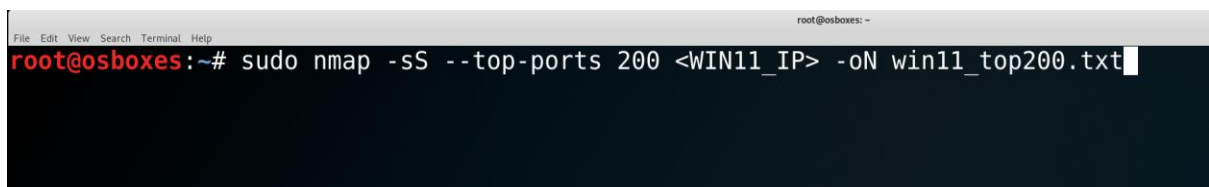
Helpful quick extraction (optional):



```
root@osboxes: ~  
File Edit View Search Terminal Help  
root@osboxes:~# grep -E "[0-9]+/tcp\s+open" metasploitable_baseline.txt | head
```

Part 4: Compare scan results against a second host (from Kali only)

From Kali, scan the IP address you identified as the Windows machine:



```
root@osboxes: ~  
File Edit View Search Terminal Help  
root@osboxes:~# sudo nmap -sS --top-ports 200 <WIN11_IP> -oN win11_top200.txt
```

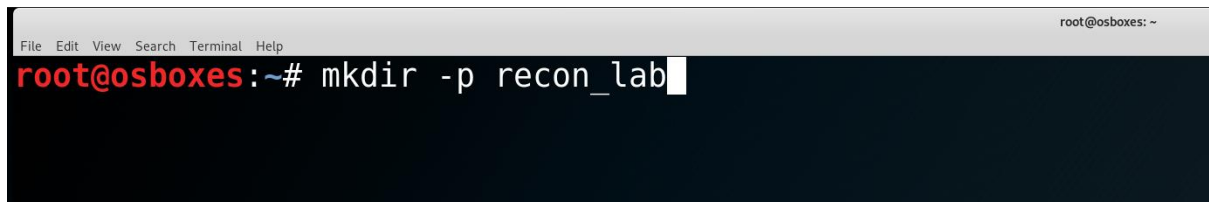
Then compare:

- Number of open ports vs Metasploitable
- Presence of filtered ports
- Differences in detected services

Write 3-4 sentences explaining how firewall behaviour changes the scan output.

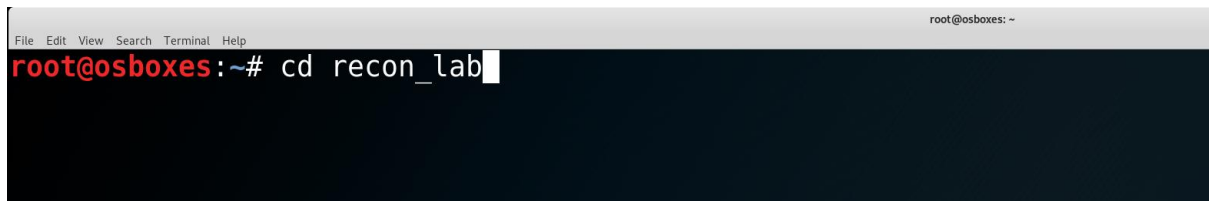
Part 5: Automate it with a Bash script

Step 1: Create a working folder



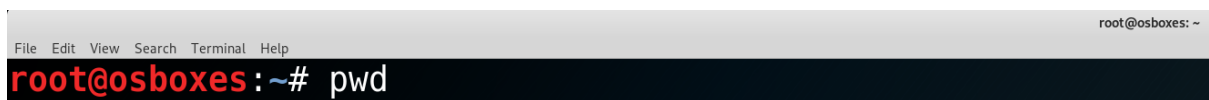
```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# mkdir -p recon_lab
```

Navigate to the folder, using the below command



```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# cd recon_lab
```

Confirm you are inside it:



```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# pwd
```

You should see something like:



```
root@osboxes:~/recon_lab# pwd
/root/recon_lab
root@osboxes:~/recon_lab#
```

Step 2: Create the script file

This is the moment the file is created.

```
root@osboxes:~/recon_lab# nano recon_pipeline.sh
```

Nano opens an empty file named `recon_pipeline.sh`.

Step 3: Paste THIS script into nano

Paste everything below exactly as shown:

```
#!/usr/bin/env bash
set -euo pipefail

# Usage: ./recon_pipeline.sh 192.168.56.1 192.168.56.50
START_IP="${1:-}"
END_IP="${2:-}"

if [[ -z "$START_IP" || -z "$END_IP" ]]; then
    echo "Usage: $0 <start_ip> <end_ip>"
    exit 1
fi

echo "[1/3] Ping sweep with fping..."
sudo fping -a -g "$START_IP" "$END_IP" 2>/dev/null | tee live_hosts.txt

echo "[2/3] Nmap scan each live host (top 200 TCP ports + service
detection)..."
mkdir -p nmap_out

while read -r host; do
```

```

[[ -z "$host" ]] && continue

echo "  Scanning $host"

sudo nmap -sS -sV --top-ports 200 "$host" -oN "nmap_out/${host}.txt"
>/dev/null

done < live_hosts.txt


echo "[3/3] Build a tiny summary report..."
REPORT="summary_report.txt"
echo "Recon Summary - $(date)" > "$REPORT"
echo "" >> "$REPORT"
echo "Live hosts:" >> "$REPORT"
cat live_hosts.txt >> "$REPORT"


echo "" >> "$REPORT"
echo "Open ports by host (from Nmap outputs):" >> "$REPORT"
for f in nmap_out/*.txt; do
    host="$(basename "$f" .txt)"
    echo "" >> "$REPORT"
    echo "== $host ==" >> "$REPORT"

    grep -E "^[0-9]+/tcp\s+open" "$f" || echo "(No open tcp ports in top
200)"
done


echo ""
echo "Done. Outputs:"
echo " - live_hosts.txt"
echo " - nmap_out/<host>.txt"
echo " - summary_report.txt"

```

Step 4: Save and exit nano

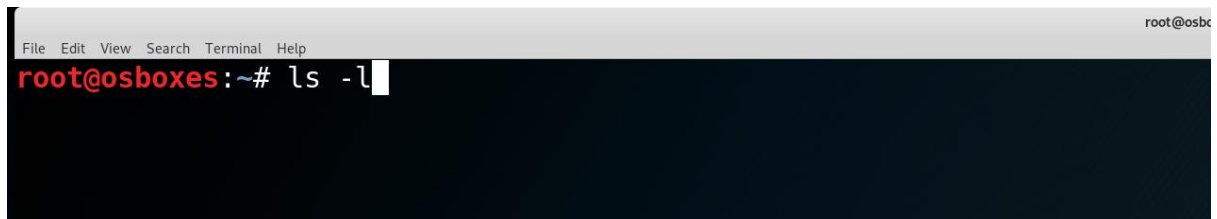
Inside nano:

- Press Ctrl + O

- Press Enter
- Press Ctrl + X

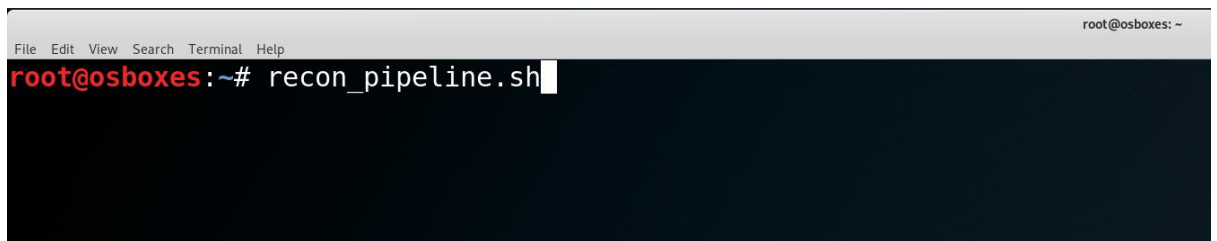
Now the file actually exists on disk.

To Confirm, type the below command:

A terminal window with a grey title bar containing 'File Edit View Search Terminal Help' and 'root@osboxes: ~' on the right. The terminal text shows 'root@osboxes:~# ls -l' with a white cursor at the end of the command.

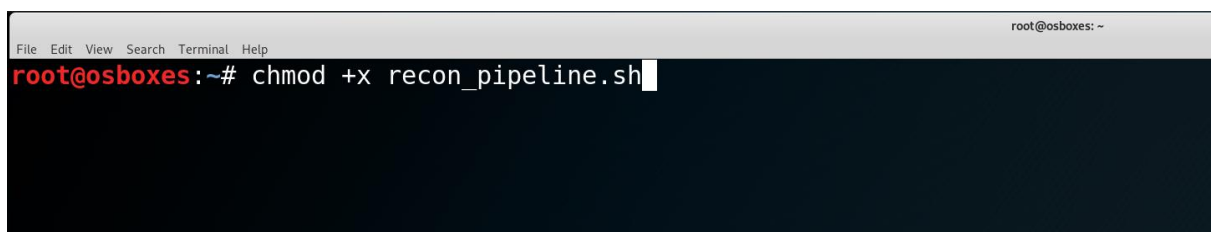
```
root@osboxes:~# ls -l
```

You should see:

A terminal window with a grey title bar containing 'File Edit View Search Terminal Help' and 'root@osboxes: ~' on the right. The terminal text shows 'root@osboxes:~# recon_pipeline.sh' with a white cursor at the end of the command.

```
root@osboxes:~# recon_pipeline.sh
```

Step 5: Make it executable

A terminal window with a grey title bar containing 'File Edit View Search Terminal Help' and 'root@osboxes: ~' on the right. The terminal text shows 'root@osboxes:~# chmod +x recon_pipeline.sh' with a white cursor at the end of the command.

```
root@osboxes:~# chmod +x recon_pipeline.sh
```

Confirm permissions:

```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# ls -l recon_pipeline.sh
```

You should see an x in the permission string, like:

```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# -rwxr-xr-x
```

Step 6: Run the script

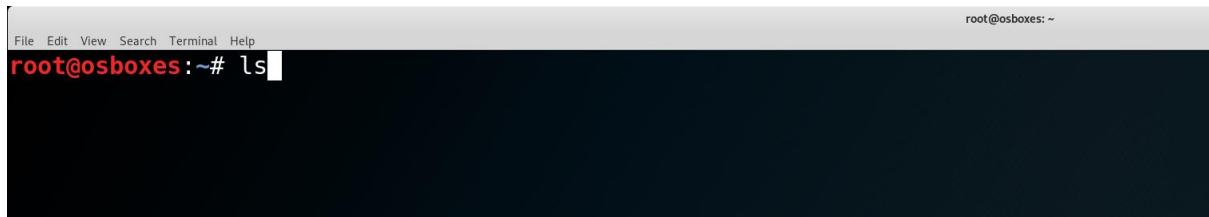
Adjust the IP range to match your subnet.

```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# ./recon_pipeline.sh 192.168.56.1 192.168.56.50
```

What happens now:

1. It runs a ping sweep
Creates: `live_hosts.txt`
2. It scans each discovered host
Creates: `nmap_out/` folder with one file per IP
3. It builds a summary
Creates: `summary_report.txt`

Step 7: Verify outputs

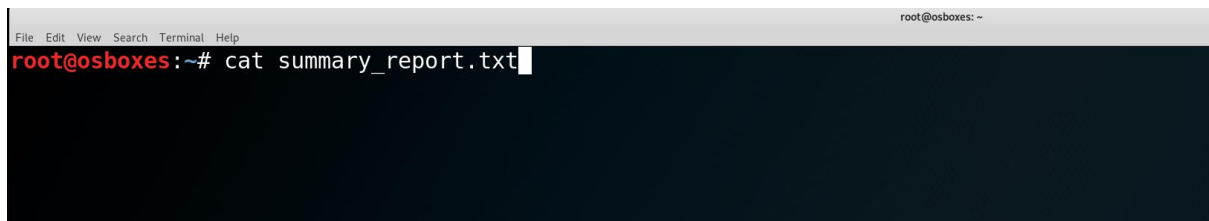


```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# ls
```

You should now see the following files:

- live_hosts.txt
- nmap_out
- summary_report.txt
- recon_pipeline.sh

Check summary:



```
File Edit View Search Terminal Help
root@osboxes: ~
root@osboxes:~# cat summary_report.txt
```

What the script is doing logically:

1. Takes two IP addresses as input
2. Uses fping to find live hosts
3. Loops through each live host
4. Runs an Nmap SYN scan with service detection
5. Extracts open TCP ports
6. Builds a clean summary report