Task 02 — Packet Sniffing, Firewall Configuration, and Vulnerability Scanning

Intern: Aliya Ansari
Date: 01-10-2025
Target / Environment: Kali Linux VM (IP 192.168.253.130), Windows VM (IP 192.168.0.206)

1. Objective

The objective of this task was to:
1. Capture and analyze network traffic using Scapy.
2. Configure a basic firewall using iptables and verify traffic control.
3. Perform vulnerability scanning using OpenVAS/GVM and document potential risks.

2. Tools and Environment

- Kali Linux VM (VMware/VirtualBox)
- Windows VM (target host)
- **Packet Sniffing:** Scapy, Python3, Matplotlib, Wireshark (optional)
- **Firewall:** iptables
- **Vulnerability Scanning:** OpenVAS / GVM (WebUI + CLI), Nmap (for quick scan evidence)

3. Part 1 — Packet Sniffing

Script: `packet_sniffer.py`

Command used:

```
sudo python3 packet_sniffer.py --iface eth0 --count 100 --timeout 30
```

Traffic generated for capture:

```
ping -c 10 8.8.8.8
curl -I http://example.com
```

3.1 Outputs Produced

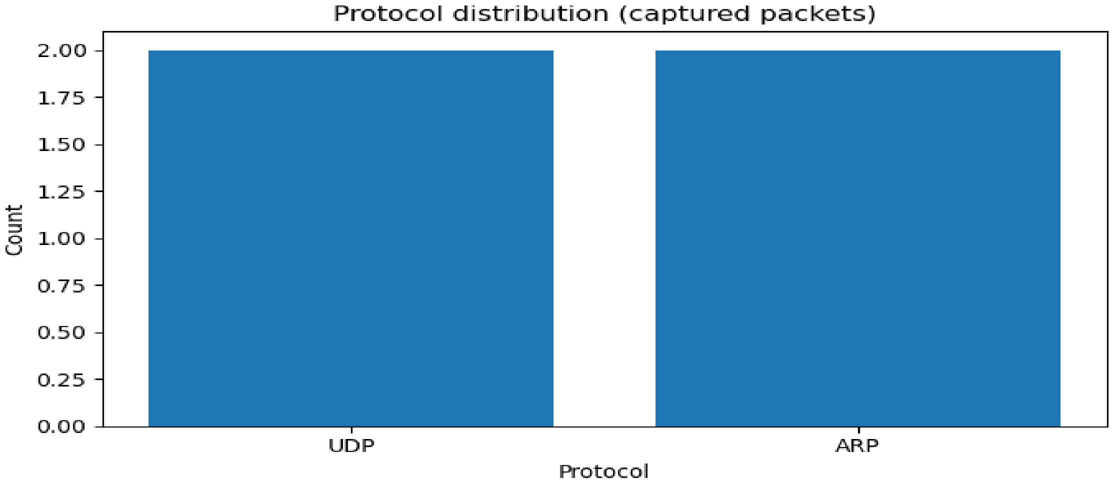
- `capture.pcap` — raw packet capture file
- `packet_sniff_report.txt` — text summary (protocol counts)
- `observed_ports.csv` — observed UDP/TCP ports
- `protocols_bar.png` — bar chart of protocol distribution

3.2 Results

Capture Summary:
Packets captured: 4 (in 30 seconds)
Protocols detected: UDP (2 packets), ARP (2 packets)

Counts Table: | Protocol | Count | |———-|——-| | UDP | 2 | | ARP | 2 |

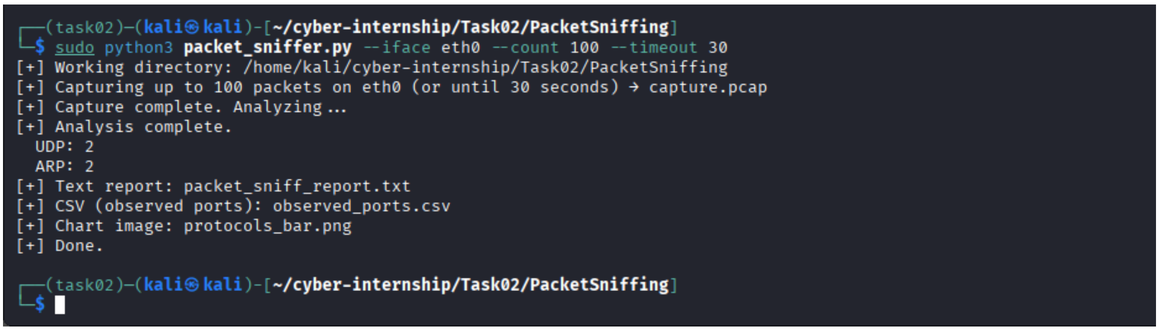
Observed Ports Table: | Host IP | Port | Protocol | |—————-|———|———-| | 192.168.253.130 | 58650 | UDP | | 13.200.20.166 | 123 | UDP |



Notes & Analysis:
- Limited packets due to short capture time.
- ARP packets show LAN address resolution activity.
- UDP packets on port 123 indicate NTP time synchronization.
- Ephemeral ports are used for outbound client connections.

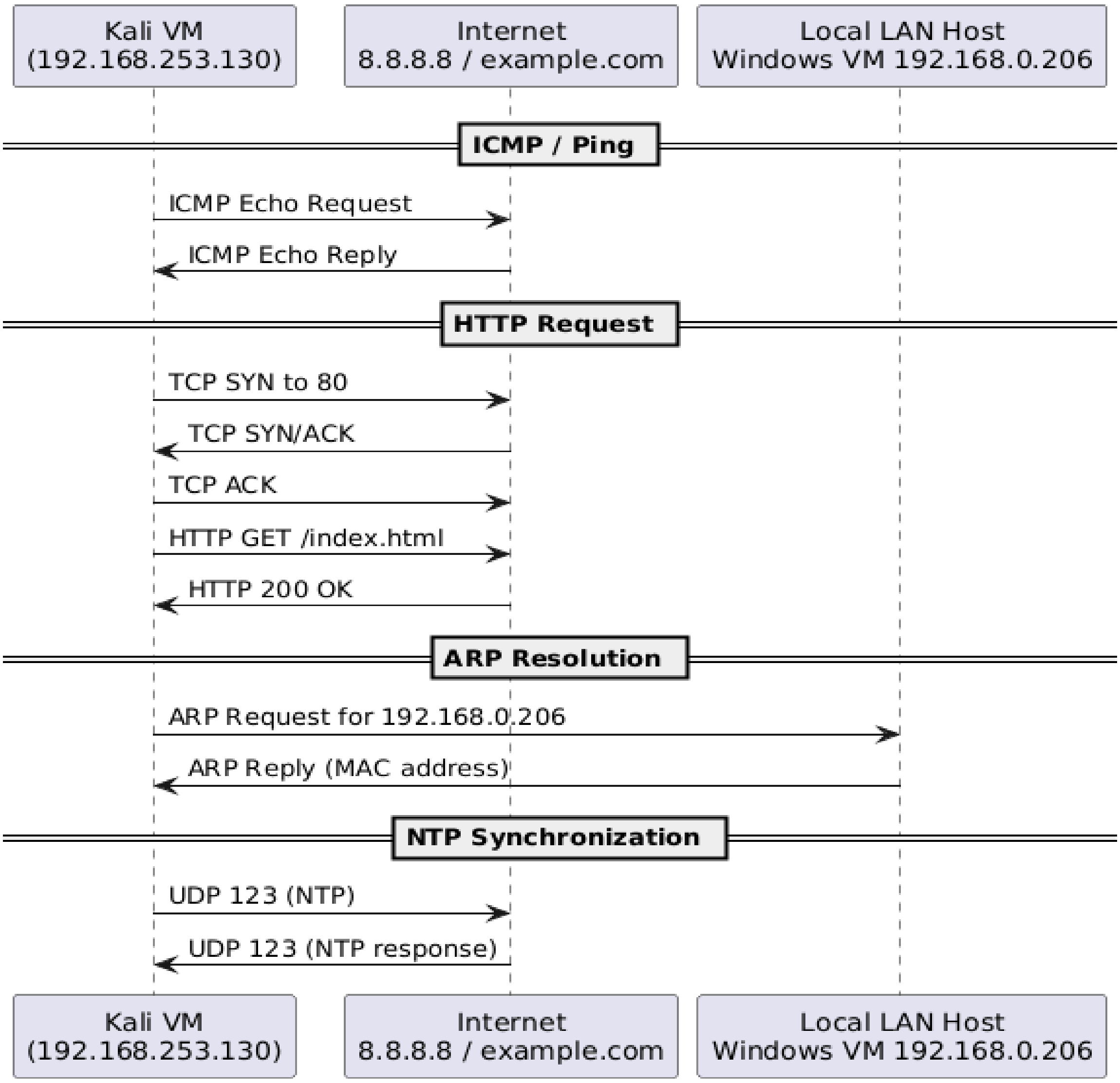
Key Learnings:
- Programmatic packet capture using Scapy.
- Protocol analysis and visualization.
- Real-world traffic observation: ARP and NTP.
- Understanding ephemeral port usage.

Terminal:



```
┌──(task02)-(kali㉿kali)-[~/cyber-internship/Task02/PacketSniffing]
└─$ sudo python3 packet_sniffer.py --iface eth0 --count 100 --timeout 30
[+] Working directory: /home/kali/cyber-internship/Task02/PacketSniffing
[+] Capturing up to 100 packets on eth0 (or until 30 seconds) → capture.pcap
[+] Capture complete. Analyzing ...
[+] Analysis complete.
  UDP: 2
  ARP: 2
[+] Text report: packet_sniff_report.txt
[+] CSV (observed ports): observed_ports.csv
[+] Chart image: protocols_bar.png
[+] Done.

┌──(task02)-(kali㉿kali)-[~/cyber-internship/Task02/PacketSniffing]
└─$ █
```

Visual Diagram:



4. Part 2 — Firewall Configuration

Objective: Set up a basic firewall using `iptables` to allow SSH/HTTP and block all other incoming traffic.

4.1 Commands Used

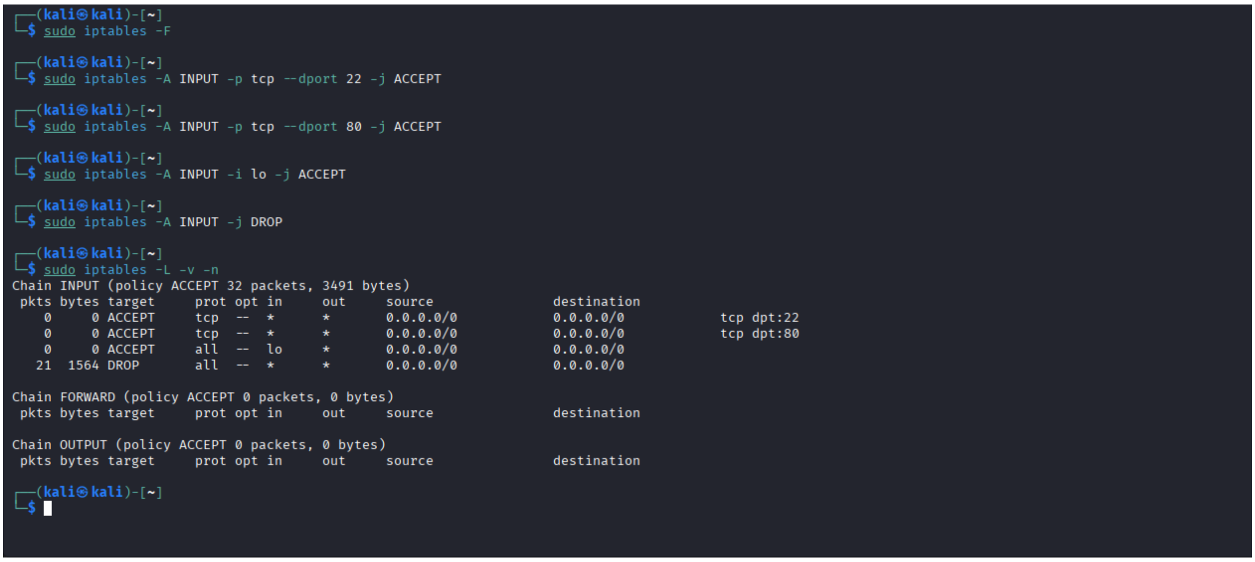
```
# Flush existing rules
sudo iptables -F

# Default policies
sudo iptables -P INPUT DROP
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT ACCEPT

# Allow loopback
sudo iptables -A INPUT -i lo -j ACCEPT

# Allow established connections
sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Allow SSH
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Allow HTTP
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# List rules
sudo iptables -L -v -n
```

4.2 Testing Results

Test	Expected	Actual
SSH (port 22)	Allowed	Connection succeeded when service running
HTTP (port 80)	Allowed	Accessible via browser
Other ports	Blocked	Connection refused or timed out

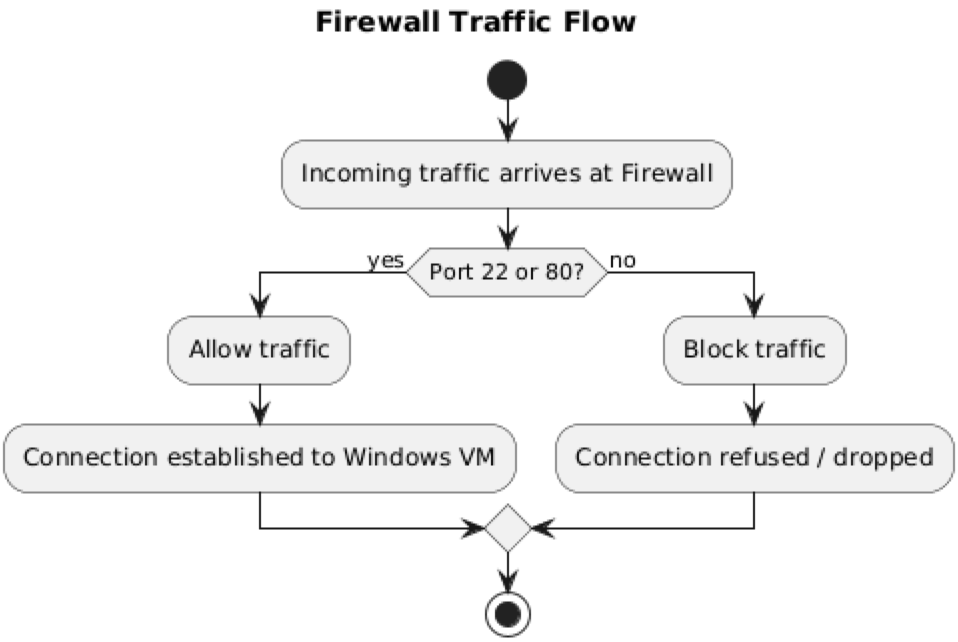
Key Learnings:
- How to define firewall rules using iptables.
- Difference between firewall rules and actual service availability.
- Testing methodology from a remote host (Windows VM) to confirm allowed/blocked traffic.

Terminal:



```
┌──(kali㉿kali)-[~]
└─$ sudo iptables -F

┌──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

┌──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

┌──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -i lo -j ACCEPT

┌──(kali㉿kali)-[~]
└─$ sudo iptables -A INPUT -j DROP

┌──(kali㉿kali)-[~]
└─$ sudo iptables -L -v -n
Chain INPUT (policy ACCEPT 32 packets, 3491 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:22
    0     0 ACCEPT     tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:80
    0     0 ACCEPT     all  --  lo     *       0.0.0.0/0            0.0.0.0/0
   21  1564 DROP       all  --  *      *       0.0.0.0/0            0.0.0.0/0

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

┌──(kali㉿kali)-[~]
└─$ ▮
```

Visual Diagram:


Firewall Traffic Flow
Incoming traffic arrives at Firewall
yes
Port 22 or 80?
no
Allow traffic
Block traffic
Connection established to Windows VM
Connection refused / dropped

5. Part 3 — Vulnerability Scanning

Objective: Perform a vulnerability assessment on the Windows VM target using OpenVAS/GVM, identify security weaknesses, and document findings.

5.1 Setup

- OpenVAS/GVM installed and running on Kali VM
- GVM services started with:

```
sudo gvm-start
```

- WebUI accessed at `https://127.0.0.1:9392`
- Target created: Windows VM (192.168.0.206)

5.2 Task Creation

- Created a new scan task in GVM:
  - **Name:** `Windows_VM_VulnScan`
  - **Target:** Windows VM
  - **Scan Config:** `Full and fast`
  - **Schedule:** Once
  - **Add results to Assets:** Yes

5.3 Scan Execution

- Started the scan from the WebUI
- Monitored progress in **Tasks → Running Tasks**

5.4 Results / Findings

Port	Service	Severity	Description / Risk	Recommended Mitigation
22	SSH	Medium	Remote login service; ensure strong credentials	Limit to trusted IPs, enable logging
80	HTTP	Medium	Web service accessible externally	Apply latest patches, enforce secure auth

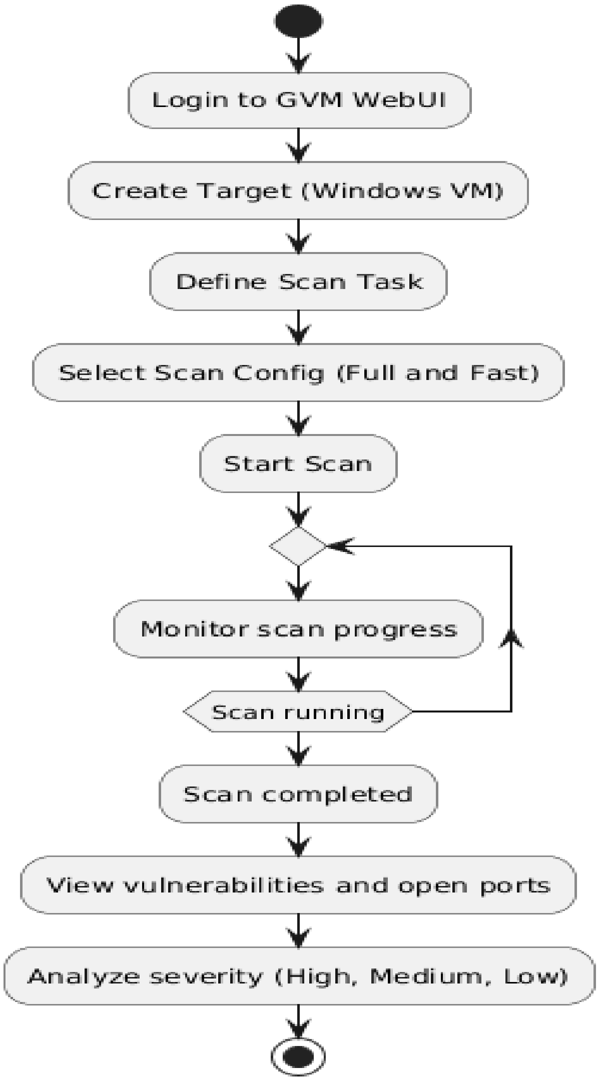
Port	Service	Severity	Description / Risk	Recommended Mitigation
135	MSRPC	High	RPC service exposed; potential for remote exploits	Restrict access, patch OS, minimize exposed services
139	NetBIOS	Medium	Legacy file-sharing protocol; information disclosure	Disable if not needed, restrict LAN access
445	SMB	High	SMB file sharing; historically targeted by ransomware	Patch SMB, block untrusted hosts, enforce strong authentication

Additional Observations:
- Scan also identified service banners and potential vulnerabilities for Oracle TNS listener and other system services.

5.5 Key Learnings / Outcomes

- Successfully created and executed a vulnerability scan on a test VM.
- Learned how to configure scan targets, choose scan profiles, and monitor scan progress in OpenVAS/GVM.
- Captured and interpreted scan results, identifying open ports, services, and potential security risks.

Visual Diagram:


Login to GVM WebUI
Create Target (Windows VM)
Define Scan Task
Select Scan Config (Full and Fast)
Start Scan
Monitor scan progress
Scan running
Scan completed
View vulnerabilities and open ports
Analyze severity (High, Medium, Low)

6. Conclusion & Key Learnings

- **Packet Sniffing:** Captured and analyzed network packets, visualized protocol distribution.
- **Firewall:** Configured iptables rules, allowed/blocked traffic, tested from a remote host.
- **Vulnerability Scanning:** Set up OpenVAS, created tasks, executed full scan, and documented open ports and potential security risks.
- **Overall:** Learned end-to-end practical workflow of network monitoring, access control, and vulnerability assessment in a lab environment.