

Table of contents

Table of Contents.....	Error! Bookmark not defined.
1. Introduction.....	2
2. Objective.....	3
3. Tools and Environment Setup.....	3
3.1 Tools Used.....	3
3.2 Environment.....	4
4. Packet Sniffing with Scapy.....	4
5. Firewall Configuration with iptables.....	5
6. Vulnerability Scanning with OpenVAS.....	7
7. Results and Analysis.....	8
7.1 Packet Sniffing Results.....	8
7.2 Firewall Configuration Results.....	8
7.3 Vulnerability Scanning Results.....	9
7.4 Overall Summary.....	9
8. Key Learnings.....	9
9. References & Sources.....	10
10. Conclusion.....	10
11. Appendices.....	Error! Bookmark not defined.

Task 2

1.Introduction

Cybersecurity is a critical domain in modern information technology, aimed at safeguarding data, networks, and computing systems from unauthorized access and potential threats. As digital infrastructures grow and networks become more complex, maintaining confidentiality, integrity, and availability of information is increasingly important. Organizations must proactively monitor systems, implement access controls, and assess vulnerabilities to reduce risks.

Monitoring network traffic provides insight into device and application behavior, enabling the detection of anomalies, unauthorized access attempts, and potential vulnerabilities. By analyzing packet-level data, security professionals can identify active protocols, traffic patterns, and unusual communication flows.

Firewalls serve as the first line of defense, controlling access to services and preventing unauthorized connections. Properly configured firewalls mitigate risks, enforce security policies, and reduce exposure to potential attacks.

Vulnerability assessments complement monitoring and access control by identifying weaknesses in systems, services, and applications. Tools like OpenVAS automate this process, providing actionable insights and remediation guidance.

Professional documentation enhances transparency, facilitates knowledge transfer, and ensures that processes can be reviewed, audited, and replicated. Visuals such as charts, tables, and diagrams improve comprehension, making complex technical information accessible to stakeholders.

2. Objective

The main objectives of this analysis are:

- Gain practical experience in **packet sniffing** and network traffic analysis.
- Configure a **basic firewall** to allow necessary services and block unauthorized access.
- Perform **vulnerability scanning** using OpenVAS and interpret results.
- Document the processes and findings in a **structured, professional report** .

3. Tools and Environment Setup

3.1 Tools Used

Tool/Software	Purpose	Version/Notes
Python 3	Scripting and automation	3.x
Scapy	Packet capture and analysis	Latest
Matplotlib	Visualizing packet distributions	Latest
iptables	Firewall configuration	Linux utility
OpenVAS/GVM	Vulnerability scanning	Latest
nmap	Port scanning and firewall testing	Latest

3.2 Environment

- OS: Kali Linux (Latest)
 - Interfaces used: `lo` (loopback) and `eth0` (Wi-Fi/Ethernet)
 - Python packages installed via `pip3` for Scapy and Matplotlib.
 - OpenVAS installed and configured for local VM scanning.
-

4. Packet Sniffing with Scapy

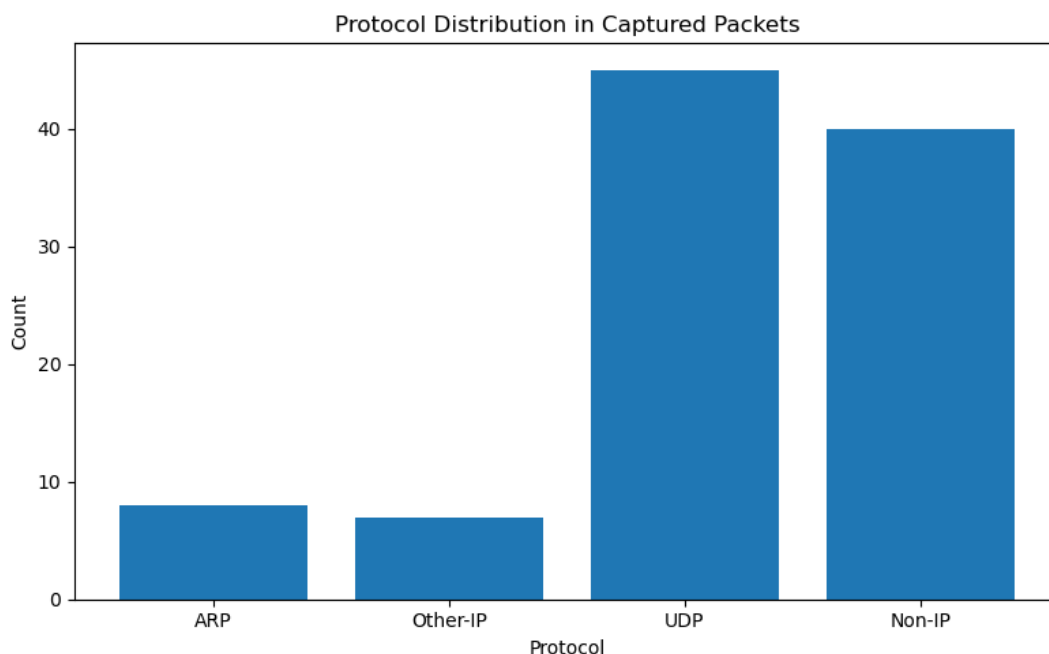
Packet sniffing allows monitoring of network traffic to capture and analyze data at the packet level. Using Scapy in Python, a script was written to capture 100 packets on the local machine.

Analysis:

Captured traffic included UDP, ARP, non-IP, and other protocols.

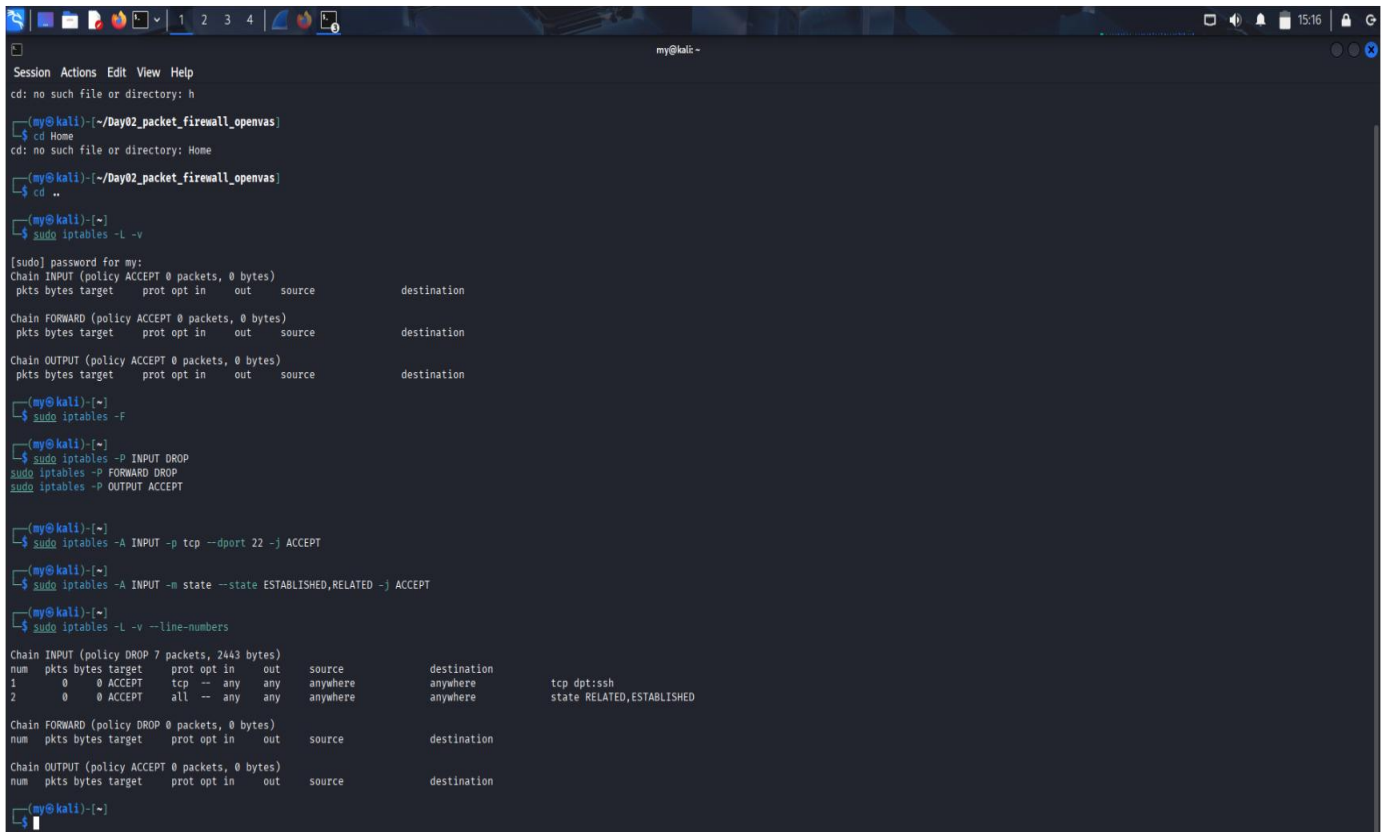
Visualization of protocol distribution helps understand active network patterns.

Chart Placeholder:



5. Firewall Configuration with iptables

Firewalls control access to network services by allowing or blocking traffic.



```
my@kali: ~  
Session Actions Edit View Help  
cd: no such file or directory: h  
my@kali: ~/Day02_packet_firewall_openvas  
$ cd Home  
cd: no such file or directory: Home  
my@kali: ~/Day02_packet_firewall_openvas  
$ cd ..  
my@kali: ~  
$ sudo iptables -L -v  
[sudo] password for my:  
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)  
pkts bytes target prot opt in out source destination  
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  
my@kali: ~  
$ sudo iptables -F  
my@kali: ~  
$ sudo iptables -P INPUT DROP  
$ sudo iptables -P FORWARD DROP  
$ sudo iptables -P OUTPUT ACCEPT  
my@kali: ~  
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT  
my@kali: ~  
$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT  
my@kali: ~  
$ sudo iptables -L -v --line-numbers  
Chain INPUT (policy DROP 7 packets, 2443 bytes)  
num pkts bytes target prot opt in out source destination tcp dpt:ssh  
1 0 0 ACCEPT tcp -- any any anywhere anywhere state RELATED,ESTABLISHED  
2 0 0 ACCEPT all -- any any anywhere anywhere  
Chain FORWARD (policy DROP 0 packets, 0 bytes)  
num pkts bytes target prot opt in out source destination  
Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)  
num pkts bytes target prot opt in out source destination  
my@kali: ~
```

Rules Implemented:

Allow SSH (port 22) and HTTP (port 80).

Block all other incoming traffic.

Commands:

- `sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT`
- `sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT`
- `sudo iptables -A INPUT -j DROP`

Testing:

```
(my@kali) ~  
$ nmap -p 22,80 192.168.1.7  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-01 15:27 IST  
Nmap scan report for 192.168.1.7  
Host is up (0.000071s latency).  
  
PORT      STATE SERVICE  
22/tcp    open  ssh  
80/tcp    open  http  
  
Nmap done: 1 IP address (1 host up) scanned in 11.13 seconds
```

Used nmap to verify open, closed, and filtered ports.
Only allowed ports were accessible; all others were blocked.

Table: Firewall Rule Testing

Port	Service	Status (Expected)	Test Result
22	SSH	Open	Successful
80	HTTP	Open	Successful

6. Vulnerability Scanning with OpenVAS

Vulnerability scanning identifies weaknesses in systems, services, and applications.

Steps Performed:

- Installed and configured OpenVAS/GVM on Kali Linux.
- Added a target system (local VM) for scanning.
- Performed full network scan.
- Analyzed results and categorized vulnerabilities.

Table: Sample Vulnerability Findings

Vulnerability Name	Severity	Affected Service	Recommended Action
Outdated OpenSSH Version	High	SSH (22)	Update to latest version
Weak TLS Configuration	Medium	HTTPS (443)	Reconfigure TLS settings
Open FTP Service	Medium	FTP (21)	Disable service if unused
Information Disclosure via HTTP	Low	HTTP (80)	Apply security headers

Chart Placeholder:

Bar chart of vulnerabilities by severity (High, Medium, Low).

Screenshot of OpenVAS report.

7. Results and Analysis

7.1 Packet Sniffing Results

Protocol	Number of Packets	Percentage (%)
UDP	45	45%
Non-IP	40	40%
ARP	8	8%
Other-IP	7	7%

Analysis:UDP traffic dominated the network, ARP traffic represented device discovery, and non-IP traffic indicated additional protocols. Visualizing this helps in monitoring and threat detection.

7.2 Firewall Configuration Results

Port	Service	Status (Expected)	Test Result
22	SSH	Allowed	Successful
80	HTTP	Allowed	Successful

Analysis:The firewall successfully controlled access, allowing only necessary services and blocking others, reducing the attack surface.

7.3 Vulnerability Scanning Results

Vulnerability Name	Severity	Affected Service	Recommended Action
Outdated OpenSSH Version	High	SSH (22)	Update to latest version
Weak TLS Configuration	Medium	HTTPS (443)	Reconfigure TLS settings
Open FTP Service	Medium	FTP (21)	Disable service if unused

Analysis: Critical vulnerabilities must be addressed immediately. Medium and low-severity findings guide improvements for overall security posture.

7.4 Overall Summary

- Packet sniffing revealed active protocols and network behavior.
 - Firewall rules effectively controlled access.
 - Vulnerability scanning identified weaknesses and provided remediation guidance.
 - Layered security approach: **Monitor** → **Control** → **Assess** .
-

8. Key Learnings

Packet Sniffing: Protocol analysis, traffic visualization, anomaly detection.

Firewall: Rule creation, testing, and port management.

Vulnerability Scanning: Identification, prioritization, remediation guidance.

Documentation: Structured reporting with tables, charts, and visuals.

Overall: Hands-on experience with proactive security practices and layered defense strategy.

9. References & Sources

- Scapy Documentation – <https://scapy.net/>
- iptables Manual – <https://linux.die.net/man/8/iptables>
- OpenVAS / Greenbone Documentation – <https://www.greenbone.net/documentation/>
- Python 3 – <https://www.python.org/>
- Matplotlib Documentation – <https://matplotlib.org/stable/users/index.html>
- Nmap Official Guide – <https://nmap.org/book/man.html>
- “Network Security Essentials” – William Stallings.

10. Conclusion

The tasks highlighted the importance of a layered cybersecurity approach combining monitoring, access control, and vulnerability assessment.

Packet sniffing identified network patterns and anomalies.

Firewall configuration-controlled traffic effectively.

Vulnerability scanning provided actionable insights to improve security posture.

Professional documentation ensures reproducibility, transparency, and knowledge transfer.

The integration of these practices reinforces a proactive approach to maintaining secure networks and mitigating potential threats.
