📄 Personal Firewall – Project Report

🔰 Title:

Personal Firewall Using Python and iptables

👨‍💻 Author:

Ram Haygrev S
Cyber Security Intern
Rajalakshmi Engineering College

🧠 Objective:

To design and implement a **lightweight personal firewall** that can:

- Monitor incoming/outgoing network packets
- Apply customizable filtering rules (block/allow IPs, ports, protocols)
- Log suspicious traffic for audit
- Automatically block malicious IPs using **iptables** (Linux-level enforcement)

🧰 Tools & Technologies Used:

Tool	Purpose
Python 3	Core language for building the firewall logic
Scapy	Packet sniffing and real-time analysis
iptables	Kernel-level firewall rule enforcement
JSON	For storing customizable firewall rules

Linux Terminal	For running the script and managing rules

📦 Project Components:

File Name	Description
`firewall.py`	Main Python script for sniffing and filtering packets
`firewall_rules.json`	JSON file to define IPs, ports, and protocols to block or allow
`firewall_log.txt`	Log file to store timestamps and packet summaries of blocked traffic

---

🛠️ Implementation Details:

1. Packet Sniffing with Scapy

- The script uses `sniff()` to capture all IP, TCP, UDP, and ICMP packets in real time.
- Captured packets are passed to a filtering function.

2. Rule Matching

- Each packet is checked against user-defined rules:
    - **Blocked IPs**
    - **Blocked or allowed ports**
    - **Blocked protocols** (e.g., ICMP)

3. Logging

- Every blocked packet is logged in `firewall_log.txt` with:
    - Timestamp
    - Reason (e.g., “Blocked Port”)

- Packet summary

4. iptables Integration
If a packet violates a rule, the source IP is dynamically blocked using:

```
sudo iptables -A INPUT -s <IP> -j DROP
```

-
- This prevents future traffic from that IP at the OS level.

5. Real-Time Monitoring

- Allowed packets are printed to the terminal.
- Blocked packets are logged silently unless explicitly printed.

🧪 Sample Rule File (`firewall_rules.json`):

```
{
  "block_ips": ["192.168.1.100", "10.0.0.5"],
  "block_ports": [23, 445],
  "allow_ports": [80, 443],
  "block_protocols": ["ICMP"]
}
```

✅ How to Run

```
# Step into the project directory
cd personal_firewall

# Run the firewall as root
sudo python3 firewall.py
```

📊 Output Example:

Console Output:

```
🛡️ Personal Firewall is running with iptables support. Press Ctrl+C to stop.
✅ Allowed: IP / TCP 10.0.0.1:56789 > 192.168.1.10:80
🚫 Blocking IP with iptables: 192.168.1.100
```

Log Entry (`firewall_log.txt`):

```
2025-07-07 14:23:15 | Blocked Port | IP / TCP 192.168.1.100:23 > 192.168.1.10:80
```

📌 Observations:

- Scapy provides detailed packet-level control and visibility.
- iptables ensures kernel-level security enforcement.
- The combination offers both **flexibility** and **strength** in filtering.
- Easily extendable to support protocols like DNS, HTTP header inspection, etc.

🚀 Possible Enhancements:

- Add a **Tkinter GUI** for real-time monitoring.
- Create a **rule editor** within the GUI.
- Implement **temporary blocking** with auto-unblock after timeouts.
- Log to **JSON or CSV** for easier reporting/analytics.

- Add **alert notifications** (e.g., popup, sound, or email).

🔐 Conclusion:

This project demonstrates how Python and Linux-native tools can be used together to build a customizable and intelligent firewall. It balances **real-time traffic monitoring** with **kernel-level enforcement**, making it suitable for personal use or in cybersecurity labs.

📎 Appendices

🧩 Dependencies:

```
pip install scapy
sudo apt install iptables
```

📚 References:

- Scapy Documentation
- iptables Man Page
- OWASP Packet Filtering Best Practices