📄 Personal Firewall in Python – Project Report

1️⃣ Introduction

Cybersecurity relies heavily on network traffic monitoring and control. Firewalls are essential tools for protecting systems from unauthorized access and malicious traffic. This project presents a **lightweight personal firewall** built in Python that captures, analyzes, and filters network packets in real-time. It allows users to define rules to allow or block traffic, logs all events for auditing, and optionally enforces rules at the system level using iptables.

2️⃣ Abstract

The project implements a Python-based personal firewall using **Scapy** for packet sniffing, a rule engine for traffic filtering, and optional interfaces for user interaction. The firewall monitors both inbound and outbound traffic, makes decisions based on configurable rules (IP, port, protocol, direction), logs all decisions for review, and optionally integrates with the system's iptables to enforce blocks. A **Tkinter GUI** allows live monitoring, while a **Flask-based web dashboard** provides a modern interface for managing rules and viewing logs. This project demonstrates the practical implementation of traffic filtering and rule-based security in a Linux environment.

3️⃣ Tools Used

Tool	Purpose
Python 3	Core programming language
Scapy	Packet sniffing and analysis
PyYAML	Load rules from YAML configuration
Tkinter	Optional GUI for live monitoring
Flask	Optional web dashboard interface
iptables	Optional system-level packet blocking
Kali Linux / Linux VM	Development and testing environment

4️⃣ Steps Involved in Building the Project

Step 1: Environment Setup ✅

- Install Python 3 and dependencies: scapy, pyyaml, tkinter.
- For web dashboard: install flask.
- Run as **root** for packet sniffing and iptables enforcement.

Step 2: Define Rules ⚙️

- Create rules.yaml with ordered rules (first-match-wins).
- Specify action, direction, protocol, src_ip, dst_ip, src_port, dst_port, and description.
- Set default_action for unmatched traffic.

Step 3: Implement Firewall Engine 🔥

1. **Sniffer**: Capture packets on the selected network interface.
2. **Metadata Extraction**: Parse protocol, source/destination IPs, ports, packet length, and direction.
3. **Rule Engine**: Match packets sequentially against rules.
4. **Logging**: Record all ALLOW/BLOCK decisions in firewall.log.
5. **Optional Enforcement**: Append DROP rules to iptables for blocked packets.

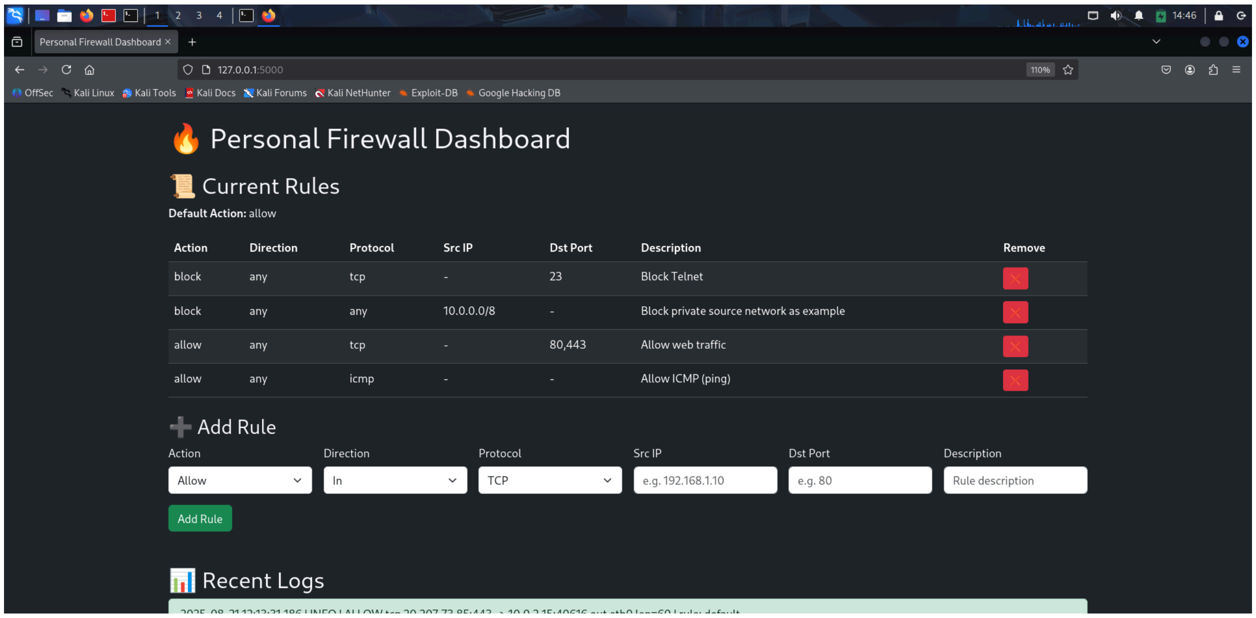
Step 4: Build GUI (Optional) 🎨

- **Tkinter GUI**: Start/stop firewall and view live logs.
- **Flask Dashboard**: Web interface for rule management and live log view.

Step 5: Testing ✅

- Block/allow specific ports (e.g., Telnet vs HTTP/HTTPS).
- Block specific IP addresses.
- Verify logs and optional iptables rules.
- Ensure default-action works as intended.

5 Screenshot



Personal Firewall Dashboard

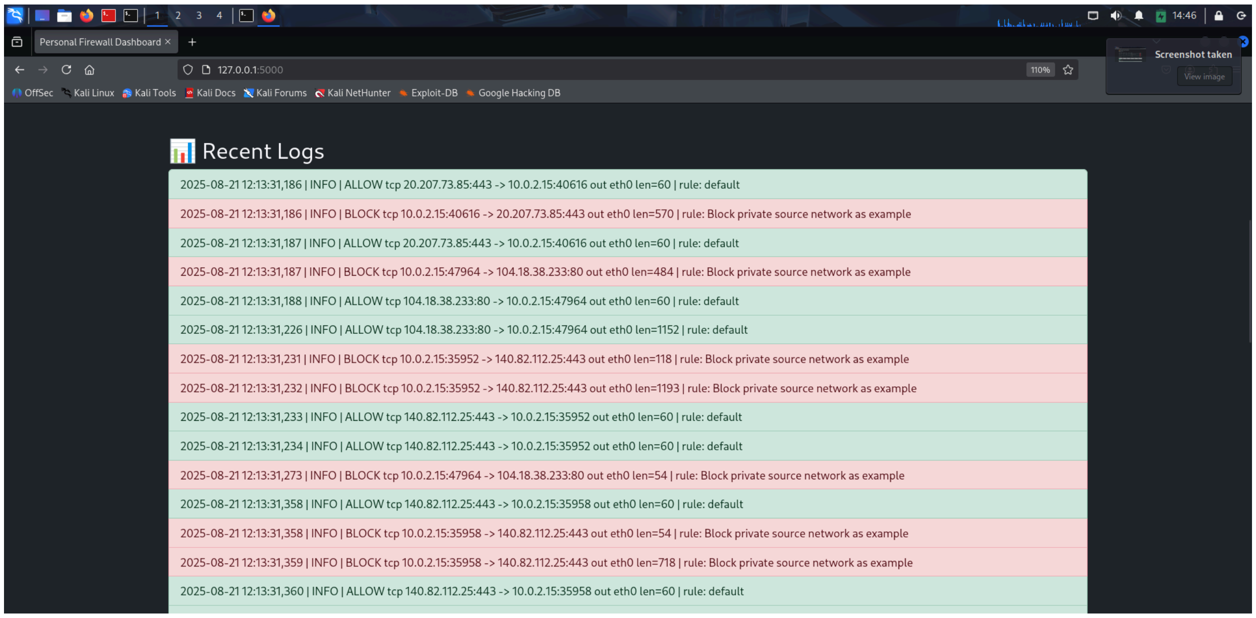
Current Rules

Default Action: allow

Action	Direction	Protocol	Src IP	Dst Port	Description	Remove
block	any	tcp	-	23	Block Telnet	
block	any	any	10.0.0.0/8	-	Block private source network as example	
allow	any	tcp	-	80,443	Allow web traffic	
allow	any	icmp	-	-	Allow ICMP (ping)	

Add Rule

Action	Direction	Protocol	Src IP	Dst Port	Description
Allow	In	TCP	e.g. 192.168.1.10	e.g. 80	Rule description



Recent Logs

```
2025-08-21 12:13:31,186 | INFO | ALLOW tcp 20.207.73.85:443 -> 10.0.2.15:40616 out eth0 len=60 | rule: default
2025-08-21 12:13:31,186 | INFO | BLOCK tcp 10.0.2.15:40616 -> 20.207.73.85:443 out eth0 len=570 | rule: Block private source network as example
2025-08-21 12:13:31,187 | INFO | ALLOW tcp 20.207.73.85:443 -> 10.0.2.15:40616 out eth0 len=60 | rule: default
2025-08-21 12:13:31,187 | INFO | BLOCK tcp 10.0.2.15:47964 -> 104.18.38.233:80 out eth0 len=484 | rule: Block private source network as example
2025-08-21 12:13:31,188 | INFO | ALLOW tcp 104.18.38.233:80 -> 10.0.2.15:47964 out eth0 len=60 | rule: default
2025-08-21 12:13:31,226 | INFO | ALLOW tcp 104.18.38.233:80 -> 10.0.2.15:47964 out eth0 len=1152 | rule: default
2025-08-21 12:13:31,231 | INFO | BLOCK tcp 10.0.2.15:35952 -> 140.82.112.25:443 out eth0 len=118 | rule: Block private source network as example
2025-08-21 12:13:31,232 | INFO | BLOCK tcp 10.0.2.15:35952 -> 140.82.112.25:443 out eth0 len=1193 | rule: Block private source network as example
2025-08-21 12:13:31,233 | INFO | ALLOW tcp 140.82.112.25:443 -> 10.0.2.15:35952 out eth0 len=60 | rule: default
2025-08-21 12:13:31,234 | INFO | ALLOW tcp 140.82.112.25:443 -> 10.0.2.15:35952 out eth0 len=60 | rule: default
2025-08-21 12:13:31,273 | INFO | BLOCK tcp 10.0.2.15:47964 -> 104.18.38.233:80 out eth0 len=54 | rule: Block private source network as example
2025-08-21 12:13:31,358 | INFO | ALLOW tcp 140.82.112.25:443 -> 10.0.2.15:35958 out eth0 len=60 | rule: default
2025-08-21 12:13:31,358 | INFO | BLOCK tcp 10.0.2.15:35958 -> 140.82.112.25:443 out eth0 len=54 | rule: Block private source network as example
2025-08-21 12:13:31,359 | INFO | BLOCK tcp 10.0.2.15:35958 -> 140.82.112.25:443 out eth0 len=718 | rule: Block private source network as example
2025-08-21 12:13:31,360 | INFO | ALLOW tcp 140.82.112.25:443 -> 10.0.2.15:35958 out eth0 len=60 | rule: default
```

6 Conclusion

The Personal Firewall project demonstrates the integration of Python, Scapy, and iptables to implement real-time traffic monitoring and filtering. It provides a hands-on understanding of packet sniffing, rule-based security, and logging. Optional GUI and web interfaces improve usability and monitoring capabilities. This project serves as a learning tool for cybersecurity enthusiasts and as a foundational prototype for more advanced personal firewall solutions.