PROJECT REPORT

TITLE: Personal Firewall using Python

Author: Mayuri Sawle

Date: 08-09-2025

Mentor/Supervisor: Elevate Labs

1. Introduction

In today's interconnected digital environment, securing systems from unauthorized access and malicious traffic is critical. A firewall acts as the first line of defense by monitoring and controlling incoming and outgoing network traffic based on predefined security rules.

This project focuses on developing a lightweight personal firewall in Python. The firewall is capable of monitoring packets, applying custom rules for blocking/allowing traffic, logging suspicious activity, and optionally enforcing system-level blocking using iptables. Additionally, a graphical user interface (GUI) has been implemented using Tkinter for live monitoring.

---

2. Objectives

The main objectives of this project are:

- To design and implement a personal firewall using Python.
- To sniff incoming and outgoing packets using Scapy.
- To define customizable rule sets for filtering traffic based on:
    - Source/Destination IP addresses
    - Ports
    - Protocols
- To log suspicious or blocked packets for auditing.
- To optionally integrate with iptables for real system-level enforcement (Linux).
- To provide a Tkinter-based GUI for live monitoring and interactive rule management.

---

3. Tools and Technologies

- Programming Language: Python 3.x
- Libraries/Modules:
    - **Scapy:** For packet sniffing and analysis
    - **Tkinter:** For GUI (optional)
    - **os / subprocess:** For executing iptables commands
    - **logging:** For activity logging
- System Utility: iptables (Linux firewall)

- Operating System: Linux (Ubuntu/Kali/Debian recommended)

---

4. Methodology

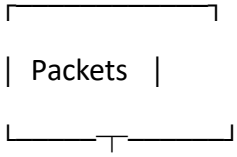
The firewall was developed step by step as follows:

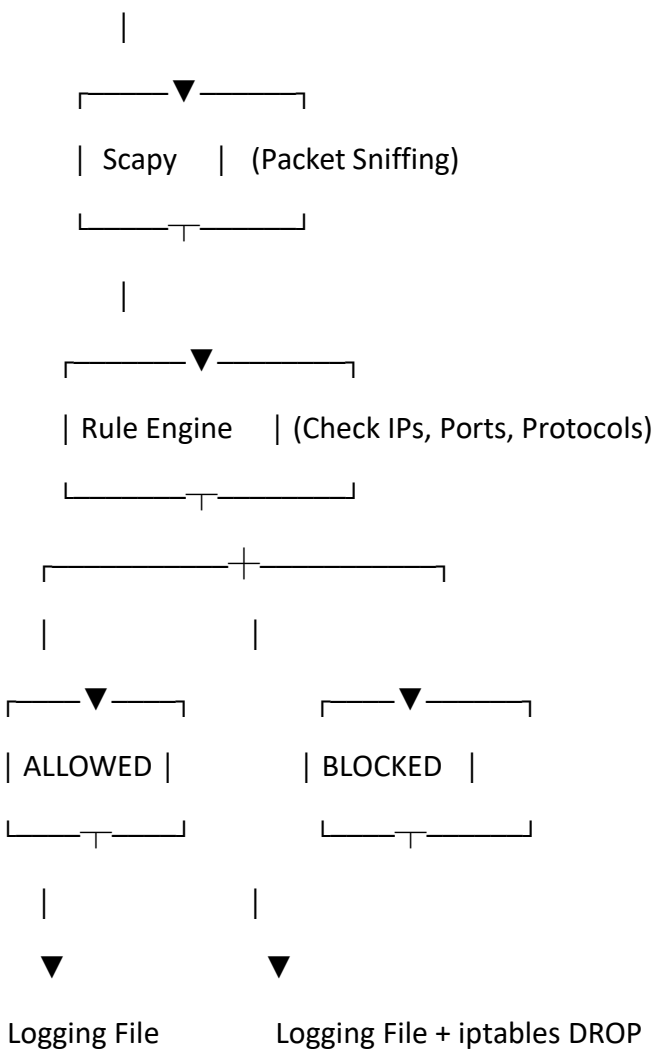
1. **Packet Sniffing**
    - Used Scapy to capture live packets.
    - Displayed packet summaries in real-time.
2. **Rule Definition**
    - Rules were defined in a JSON configuration file (rules.json).
    - Example rules include blocked IPs, blocked ports (e.g., SSH/Telnet), and blocked protocols (e.g., ICMP).
3. **Rule Application & Filtering**
    - Each captured packet is checked against the rule set.
    - If a rule matches, the packet is marked as **BLOCKED**; otherwise, it is **ALLOWED**.
4. **Logging**
    - All packet actions are logged into firewall.log with timestamps for auditing.
5. **System-Level Enforcement (Optional)**
    - Using iptables, matching packets are dropped at the kernel level.
    - The script creates a dedicated chain (FW_PY) to avoid interfering with other rules.
6. **Graphical User Interface (Optional)**
    - Tkinter-based GUI displays live packets.
    - Provides an option to add a blocked IP dynamically.

---

5. System Architecture

Workflow Diagram:





---

6. Implementation

- **Configuration File (rules.json)**

```
{
  "block_ips": ["192.168.1.128"],
  "block_ports": [22, 23],
  "block_protocols": ["ICMP"]
}
```

- **Execution Commands**
    - Run in CLI mode:
    - sudo python3 firewall.py --config rules.json
    - Run with iptables enforcement:

- sudo python3 firewall.py --config rules.json --iptables
- Run GUI:
- sudo python3 firewall.py --config rules.json --gui --iptables

---

7. Results

- Successfully captured and analyzed packets using Scapy.
- Implemented a rule-based filtering engine.
- Logged all packet activity with clear status (ALLOWED / BLOCKED).
- Integrated iptables to enforce real-time blocking at the system level.
- Developed a Tkinter GUI for live monitoring and interactive rule addition.

---

8. Conclusion

The project successfully demonstrates how a personal firewall can be built using Python. It provides both monitoring and enforcement features, along with a GUI for ease of use.

Key achievements:

- Lightweight and customizable firewall.
- Effective logging of suspicious activity.
- Optional integration with system-level blocking (iptables).
- Extendable architecture (new rules or protocols can be added easily).

---

9. Future Enhancements

- Add support for allow-list rules (whitelisting).
- Provide a web-based dashboard for remote monitoring.
- Implement signature-based intrusion detection.
- Add real-time alerts via email or Telegram.
- Cross-platform compatibility (Windows firewall integration).

---

10. Reflection

This project enhanced my understanding of:

- Network packet structure (IP, TCP/UDP, ICMP).
- Firewall concepts (rules, filtering, logging).
- Practical use of Scapy and iptables for security tasks.
- GUI design in Python with Tkinter.
- The importance of logging and auditing in security tools
