Personal Firewall Project Report

Introduction

In today's digital landscape, cybersecurity is paramount. A personal firewall acts as a barrier between a user's device and potential threats from the internet or local network. This project implements a **Python-based personal firewall** with both **GUI and CLI modes**, offering real-time packet filtering, logging, and rule-based traffic control.

Abstract

The firewall leverages **Scapy** for packet sniffing and filtering, **Tkinter** for a user-friendly interface, and **psutil** for process monitoring. Key features include:

- **Rule-based filtering** (IP, port, protocol blocking)
- **Real-time traffic monitoring** with logging
- **Process and service identification** for network connections
- **Interactive GUI** with status indicators and log management

---

Tools Used

1. **Python 3.x** – Core programming language
2. **Scapy** – Packet manipulation and sniffing
3. **Tkinter** – GUI development
4. **psutil** – Process and network monitoring
5. **Logging module** – Event tracking and storage
6. **Threading** – Concurrent packet sniffing and GUI updates

---

Steps Involved in Building the Project

1. Packet Sniffing & Filtering

- Used **Scapy** to capture and analyze network packets.
- Implemented a **rule engine** (RULES dictionary) to block/allow traffic based on:

- **IP addresses** (blacklist)
- **Ports** (e.g., blocking Telnet on port 23)
- **Protocols** (allow only TCP/UDP by default).

2. Logging & Process Tracking

- Logged filtered packets with **timestamps**, **source/destination IPs**, and **ports**.
- Mapped ports to services (e.g., 80 → HTTP) using a predefined dictionary.
- Used **psutil** to identify applications associated with network connections.

3. GUI Development

- Built an intuitive dashboard using **Tkinter** with:
  - **Traffic Monitor** (real-time logs)
  - **Rule Management** (add/remove IPs, ports)
  - **Start/Stop firewall** controls.
- Applied a **modern UI theme** with status indicators (active/inactive).

4. Multithreading

- Separated **packet sniffing** (background thread) from the **GUI main loop** to prevent freezing.
- Used a **Queue** to safely pass log messages between threads.

Conclusion

This project demonstrates how a **Python-based firewall** can provide **real-time network protection** with customizable rules. Future enhancements could include:

- **Deep packet inspection** (block malicious payloads).
- **Cloud sync** for rule management across devices.
- **Automated threat detection** using machine learning.

The combination of **Scapy for packet analysis** and **Tkinter for GUI** makes this a versatile tool for both **beginners learning networking** and **users seeking lightweight protection**.