Personal Firewall using Python:

**Abstract:**

This project implements a lightweight personal firewall in Python that filters incoming and outgoing network traffic based on customizable rules. It uses *scapy* for packet sniffing, *SQLite* for persistent logging, and allows interactive rule management from the command line. The firewall can block or allow traffic based on IP addresses, ports, and protocols. Suspicious traffic is logged for later review, providing an easy-to-use security solution for personal systems.

**Introduction:**

A firewall is a security mechanism that monitors and controls network traffic based on predefined rules. Traditional firewalls are often integrated into operating systems or external hardware, but this project demonstrates how a simplified personal firewall can be developed using Python. The purpose of this system is to help users gain a fundamental understanding of network traffic filtering and security monitoring.

This personal firewall is designed for Linux environments and leverages *Scapy* for real-time packet interception. It allows administrators to add or modify rules dynamically, stores logs of all traffic decisions, and provides statistics on allowed and blocked packets.

**Tools Used:**

* Python 3: Core language for implementation.
* Scapy: Library for packet sniffing, analysis, and manipulation.
* SQLite3: Lightweight relational database for logging packet metadata and rules.
* Threading: Enables packet sniffing to run continuously without blocking the user interface.
* Iptables (optional): Can be integrated to enforce rules at the Linux kernel level.
* Tkinter (optional): Provides the possibility of building a GUI for live monitoring.

**Steps Involved in Building the Firewall**

1. Database Setup  
   An SQLite database was created with two tables:
   * rules: Stores firewall rules such as IP, port, action (allow/block), and description.
   * packets: Logs details of all processed packets with source, destination, protocol, port, and action taken.
2. Rule Management
   * Predefined rules are loaded at startup (e.g., block TCP port 22, allow HTTP/HTTPS traffic).
   * Users can add new rules via the CLI (rule type, value, and action).
   * Rules are applied to both incoming and outgoing packets.
3. Packet Sniffing
   * Using Scapy, all IP packets are captured in real time.
   * Relevant details (IP addresses, ports, protocol) are extracted for further analysis.
4. Packet Checking & Action Enforcement
   * Each packet is checked against the loaded rule set.
   * If a match is found (e.g., IP or port match), the corresponding action (allow/block) is applied.  
     -If no match is found, the packet is allowed by default.
5. Logging & Statistics
   * Each packet’s metadata is inserted into the SQLite database along with the decision (allowed/blocked) and reason.
   * Runtime counters provide total allowed, blocked, and processed packets.
6. User Interface (CLI)  
   A simple menu-driven CLI allows users to:
   * Start or stop the firewall.
   * Show or add rules.
   * Display live statistics of network traffic.

**Conclusion:**

The project successfully demonstrates how Python can be used to build a functional personal firewall. It provides packet sniffing, rule-based filtering, logging, and user interaction within a lightweight framework. While this implementation works at a user level, integrating *iptables* enforcement could make it more secure and production-ready. The system can be extended further by adding a Tkinter-based GUI for visual monitoring, exporting logs in different formats, and integrating intrusion detection features.

This project highlights how easily customizable network security tools can be built using open-source libraries and serves as a solid foundation for future development in cybersecurity applications.