Personal Firewall Using Python

GitHub: https://github.com/OishiRakshit/Personal_firewall

Overview

This project presents a personal firewall developed using Python and Linux-based tools to monitor and control network traffic. Designed with a focus on simplicity and security, it allows users to block specific IP addresses and ports through a graphical interface (GUI).

This firewall solution demonstrates fundamental cybersecurity practices and Python programming capabilities suitable for practical deployment in personal or small office environments.

Tools Used

- Python – The backbone of modern automation and scripting in cybersecurity.
- Scapy – A powerful packet crafting and sniffing library for network analysis.
- iptables – Linux's native firewall utility for packet filtering and traffic control.
- Tkinter – Python's built-in GUI library for lightweight desktop applications.
- Kali Linux – An advanced penetration testing OS tailored for security professionals.

Directory

The project is built around six major Python modules:

```
Personal_firewall/
|── firewall.py           # Core firewall engine that monitors network traffic
|── rules.json            # Configuration file defining custom allow/block rules
|── logger.py             # Module responsible for event logging and log formatting
|── main.py               # Launches the graphical user interface for user interaction
|── iptables_blocker.py   # Manages system-level packet blocking via iptables comd.
|── firewall_log.txt      # Log file storing real-time records of network activity
```

Key Functionalities

- Real-time packet sniffing with Scapy
- Rule-based filtering on IP, port and protocols (TCP, UDP, ICMP)
- Automatic log generation with timestamped entries
- Tkinter-based GUI interface with live log display
- Dynamic iptables rule injection and flushing
- Modular architecture allowing easy rule customization

Workflow

1. **Real-Time Traffic Monitoring:**

   The firewall.py script initiates live packet capture using the Scapy library, continuously monitoring incoming network traffic.

2. **Rule-Based Evaluation:**

   Each captured packet is analysed and compared against predefined rules stored in the rules.json configuration file.

3. **Action Execution based on Rules:**

   If a packet meets any blocking criteria (e.g., access on port 80), the system will:

   - Enforce the block through iptables at the OS level
   - Record the event in a log file for audit and traceability

4. **User Control via Interface:**

   The GUI, powered by Tkinter, provides users with simple controls to start or stop the firewall and monitor traffic logs in real time.

Development Challenges and Solutions

- **Packet Capture Fails Without Root:**
  Resolved by ensuring the tool runs with elevated privileges and explicitly specifying active interfaces.
- **Log Display Latency in GUI:**
  Solved using Tkinter's ‘after()’ method to refresh the log area at regular intervals.
- **iptables Conflicts with System Defaults:**
  Introduced dedicated flushing logic before applying custom rules to prevent overlapping.

Testing and Validation

- ➢ The firewall was rigorously tested on Kali Linux.
- ➢ Sample traffic was generated using network scanning tools and browser-based access to test rule effectiveness.
- ➢ Rule updates were validated dynamically through GUI interaction and manual editing of the rules.json file.

Conclusion

This Python personal firewall efficiently manages network traffic through both command-line and GUI interfaces. The project enhanced my understanding of Linux networking, packet filtering, GUI design, and real-world Python application development.

Future iterations will integrate rule learning algorithms and email threat notifications for improved usability and automation.