Personal Firewall using Python

1. Introduction

In the current era of increasing cybersecurity threats, monitoring and controlling network activity is essential to protect systems from unauthorized access and malicious behavior. Firewalls play a crucial role in filtering network traffic and enforcing security policies. This project aims to simulate a personal firewall using Python. The goal is to build a simple, customizable, and lightweight firewall that monitors real-time connections and flags or logs suspicious traffic based on defined rules for IP addresses and ports.

2. Abstract

The project involves the development of a command-line based personal firewall tool using Python. It lists all active network connections on a system using the psutil library, identifies potentially harmful connections based on user-defined blacklists (IPs and ports), and logs those entries with timestamps. This project provides a basic understanding of how packet filtering and connection monitoring work, and forms a foundation for building more advanced intrusion detection or prevention tools in the future. A simple version of the firewall was developed for Windows systems, but it can be extended for Linux with iptables integration and GUI support using Tkinter.

3. Tools and Technologies Used

- **Python 3.8** – Programming language used to develop the firewall
- **psutil** – Python library used to retrieve system and network connection details
- **socket** – For identifying protocols (TCP/UDP)
- **datetime** – Used to timestamp and log blocked entries
- Tkinter – For future GUI support*(Optional)*
- iptables – For system-level blocking on Linux*(Optional)*

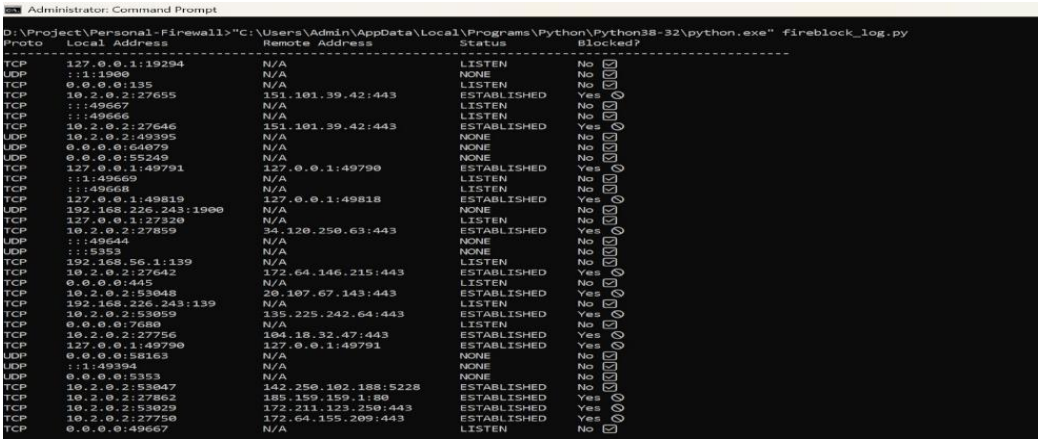
4. Steps Involved in Building the Project

1. **Connection Monitoring**: Used psutil.net_connections() to list all active TCP and UDP connections, along with their local and remote addresses, status, and protocols.
2. **Custom Blocklist Setup**: Defined a list of suspicious IP addresses and commonly targeted ports such as HTTP (80) and HTTPS (443), which were checked against all active connections.
3. **Detection and Status Display**: For each connection, the tool checked if the remote IP or port was in the blocklist. If so, the connection was marked as "Blocked" in the terminal output.

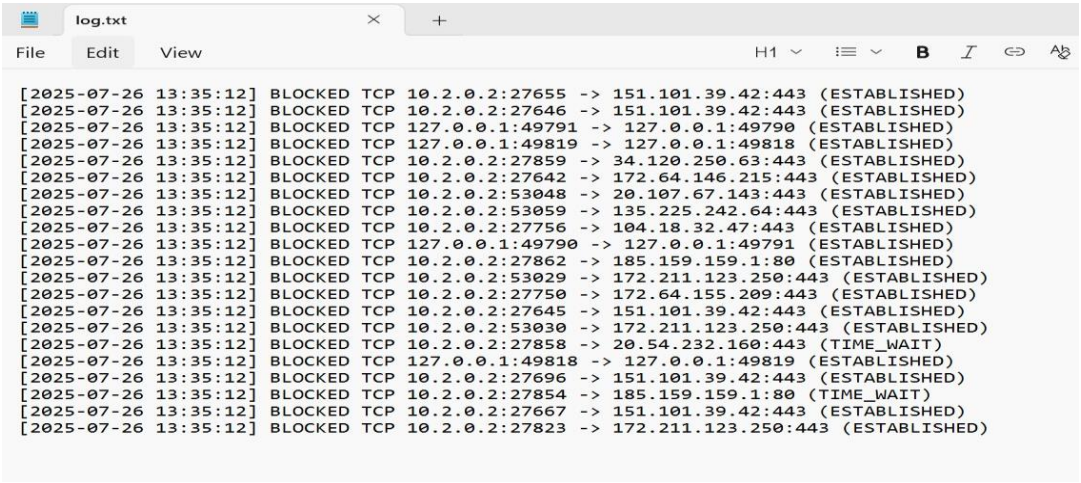
4. **Logging Suspicious Connections**: Blocked connections were logged to a file named log.txt with a timestamp, protocol, source and destination addresses, and connection status for future analysis.
5. **Future Expansion (Optional)**: The firewall can be extended with a GUI using Tkinter for live monitoring and settings control, or integrated with Linux's iptables for actual packet dropping functionality.

5. Screenshots

- **Figure 1**: Console output showing active and blocked network connections



- **Figure 2:** Log file with blocked connection entries and timestamps



6. Conclusion

This project successfully demonstrates how Python can be used to create a basic personal firewall to monitor and filter network traffic. While it does not block packets at the OS level, it serves as an educational and practical tool for understanding traffic inspection, rule-based filtering, and log management. It also provides foundational skills in scripting, network analysis, and Python automation that are essential for cybersecurity professionals.