PERSONAL FIREWALL USING PYTHON

Abstract

This project focuses on the development of a lightweight **personal firewall** using **Python**, designed to monitor, filter, and control network traffic based on defined security rules. The firewall utilizes the **Scapy** and **NetfilterQueue** libraries to inspect packets in real time and enforce rule-based filtering. It enables users to block or allow traffic based on IP address, port number, and protocol type. This implementation provides an educational demonstration of how software-level firewalls operate in Linux environments.

Introduction

A firewall acts as a barrier between trusted and untrusted networks, analyzing data packets and determining whether they should be allowed or denied. Traditional hardware firewalls are expensive and complex to configure. This project demonstrates how a **software-based firewall** can be implemented using Python to provide customizable traffic control and monitoring for personal systems. It serves both as a **security tool** and a **learning project** for understanding packet filtering and network protocols.

Tools Used

1. **Python 3** – Core programming language for implementation.
2. **Scapy** – Used for packet parsing and inspection.
3. **NetfilterQueue** – Interfaces with Linux `iptables` to capture and modify packets in user space.
4. **iptables (Linux)** – Kernel-level packet filtering framework used to redirect packets to the Python program.
5. Logging module – For recording blocked and allowed traffic events.
6. JSON – Used for storing customizable firewall rules.

Steps Involved in Building the Project

1. Requirement Analysis
Defined project scope — building a personal software firewall capable of blocking IPs, ports, and domains.

2. Environment Setup

Installed dependencies (`scapy`, `netfilterqueue`) and configured `iptables` rules to forward traffic to a queue.

3. Packet Inspection Logic
Implemented real-time packet capture using `NetfilterQueue`. Parsed IP, TCP, and UDP layers with `Scapy` to extract relevant header information.

4. Rule Management System
Created a JSON-based rule file to specify blocked IPs, ports, and domains. Added logic for rate-limiting and DNS-based blocking.

5. Logging and Monitoring
Configured a rotating log system to record all blocked or suspicious packets for audit and debugging purposes.

6. Testing and Validation
Verified by generating packets using `ping`, `telnet`, and `curl` commands, and confirmed proper blocking/allowing actions via log outputs.

Conclusion

The “Personal Firewall using Python” project successfully demonstrates how a custom firewall can be built with open-source tools and minimal resources. It provides users with better visibility and control over their network traffic while reinforcing concepts of packet filtering, rule-based access control, and network security. The project can be extended to include a **GUI for rule management**, **IPv6 support**, and **machine-learning-based anomaly detection** in future versions.