Personal Firewall with Tkinter GUI – Project Report

Abstract

This project implements a Python-based personal firewall with a Tkinter graphical interface. It provides real-time network monitoring, rule-based packet filtering, logging, and optional iptables system-level blocking. The firewall uses Scapy for packet sniffing and supports a user-friendly interface for starting, stopping, and configuring the firewall. The objective is to demonstrate the use of Python in cybersecurity, GUI development, and network traffic control.

Introduction

A personal firewall monitors and controls network traffic based on defined security rules. Traditional firewalls can be complex, but Python offers flexibility to build a lightweight, customizable firewall for learning and experimentation. This project combines Scapy-based packet sniffing with a Tkinter GUI that allows users to manage firewall actions visually.

The firewall identifies and filters packets based on IP, port, protocol, and direction, and logs events for auditing. This system is ideal for cybersecurity students and hobbyists.

Tools Used

- Python 3 — Main programming language
- Scapy — For sniffing and analyzing packets
- Tkinter — For designing the GUI
- JSON — For storing rule sets
- iptables (Linux optional) — For real packet blocking
- ReportLab — To generate this PDF
- Linux OS (Kali/Ubuntu recommended)

Steps Involved

1. Environment Setup

Python, Scapy, and Tkinter are installed. The firewall must be run with root privileges to allow packet sniffing. A default rules.json is automatically generated if missing.

2. Interface Detection

The program auto-detects the first non-loopback network interface (e.g., wlp2s0, eth0).

The user may also select an interface manually from the GUI.

3. Packet Sniffing with Scapy

The firewall uses AsyncSniffer from Scapy to capture live packets. Each packet’s source/destination IP, ports, protocol, and direction (in/out) are extracted.

4. Rule Engine

Rules are stored in JSON format and include action (allow/block/log), IPs, ports, direction, and protocol. Each captured packet is compared against the rules sequentially.

The first matching rule decides the packet's fate.

5. Logging

All events—including allowed, blocked, or logged packets—are written to firewall.log.

Each entry includes timestamp, action, IPs, ports, protocol, and rule description.

6. iptables Integration (Optional)

For real OS-level blocking, the program can insert DROP rules using iptables when a packet is blocked. This feature is optional and controlled by a GUI checkbox.

7. Tkinter GUI

The GUI includes:

• Interface selection dropdown

• Start/Stop firewall buttons

• Reload rules button

• Checkbox for enabling iptables

• Real-time log display panel

The GUI runs smoothly using background threads for sniffing and log updates.

Conclusion

This project successfully demonstrates building a functional personal firewall with packet sniffing, rule-based filtering, GUI control, and logging. It highlights Python’s strengths in

network monitoring and user interface creation. The solution is simple enough for students, yet powerful enough to illustrate core concepts of cybersecurity, packet analysis, and firewall mechanics. This project can be extended with features like rule editing in GUI, traffic graphs, and deep packet inspection.