REAL-TIME PACKET SNIFFER AND DYNAMIC FIREWALL GUI USING PYTHON

◆ INTRODUCTION

In today's digital age, network security has become a major concern. Monitoring and controlling the flow of packets across a network is essential for identifying threats and anomalies. This project presents a desktop-based real-time packet sniffer and dynamic firewall system with a graphical user interface (GUI), built using Python. It provides packet-level visibility and allows users to define blocking rules for IP addresses, ports, or protocols dynamically.

◆ ABSTRACT

This project implements a real-time packet capturing and filtering tool that helps analyze network traffic for monitoring and security purposes. It leverages the Scapy library for capturing live packets and PyQt5 for the graphical interface. The application supports protocol-based filtering (TCP, UDP, ICMP), manual and auto-updating visualizations, and custom rule-based firewalls with persistent settings. An alert system notifies users of blocked packets in real-time, with toggle controls for enabling or disabling such notifications. Users can save filtered results or adjust settings dynamically through an interactive GUI.

◆ TOOLS USED:

Component	**Technology / Library**
Programming Lang.	Python 3.x
GUI Framework	PyQt5
Packet Capture	Scapy
Data Storage	JSON
Visualization	Matplotlib
Multithreading	Python threading module

◆ STEPS INVOLVED IN BUILDING THE PROJECT

1. **Environment Setup**
   Installed Python dependencies including PyQt5, Scapy, and Matplotlib.

2. **Packet Sniffer Module**
   Implemented real-time packet capturing using Scapy, extracting protocol, IP, and port info.

3. **Firewall Rule Management**
   Designed a system to block packets based on IP, port, or protocol; rules persisted in JSON.

4. **GUI Implementation**
   Developed separate windows using PyQt5:

   - Packet Viewer (with alerts and filters)
   - Rule Manager (add/delete rules)
   - Graph Viewer (protocol distribution pie chart)

5. **Alerting System**
   Integrated a toggle-able alert system using QMessageBox to notify users of blocked packets.

6. **Visualization & Auto Refresh**
   Integrated Matplotlib into PyQt5 to display dynamic pie charts with optional auto-refresh.

7. **File Save/Load Features**
   Enabled saving of filtered packets and persistent firewall rules.

◆ CONCLUSION

The final application serves as a functional prototype of a real-time packet sniffer combined with a customizable firewall manager. With a user-friendly GUI, it allows both novice and technical users to analyze and control network traffic easily. Features like alerts, graphing, and persistent rules offer both insight and security. This tool can be extended in the future for advanced functionalities like traffic shaping, anomaly detection, and intrusion prevention.