Network Packet Sniffer with Alerts - Project Report

Abstract

This project involves developing a real-time network packet sniffer with anomaly detection and alerting capabilities. The sniffer captures live network traffic, analyzes packet headers, detects suspicious behaviors such as port scanning and flooding, stores traffic data in a SQLite database, and alerts the user on threshold breaches. Additionally, a graphical user interface (GUI) visualizes traffic distribution and allows exporting captured data for further analysis.

Introduction

Network monitoring is vital for identifying unusual activity that could indicate security threats. Packet sniffers capture network packets to analyze traffic patterns and detect anomalies. This project implements a packet sniffer in Python using Scapy for packet capture, SQLite for data storage, and Tkinter for GUI. It offers basic intrusion detection features by monitoring connection patterns and volume.

Tools Used

- Python 3: Core programming language for implementation
- Scapy: Packet capturing and network analysis library
- SQLite: Lightweight database for storing captured packet data
- Tkinter: GUI toolkit for Python to build a user-friendly interface
- Matplotlib: For live pie-chart visualization of protocol usage
- CSV module: To export captured data for external use

Steps Involved in Building the Project

1. Environment Setup: Installed Python and required libraries (Scapy, Matplotlib, Tkinter).
2. Packet Capture: Used Scapy to capture packets live from a selectable network interface.
3. Data Storage: Designed and implemented a SQLite database schema to log packet details (timestamp, source/destination IP and ports, protocol, length, flags).
4. Anomaly Detection: Created simple detection algorithms for port scanning (multiple destination ports from same IP) and flooding (high packet rate in short time). Alerts are logged for suspicious activity.
5. Graphical Interface: Built a Tkinter-based GUI with controls to start/stop sniffing, select the network interface, view live protocol distribution pie chart, and export data to CSV.

6. Testing: Verified capturing traffic from the chosen interface, triggering alerts by simulating port scans and flooding, and exporting logs correctly.

Conclusion

This project demonstrates a functional packet sniffing tool with basic intrusion detection and alert capabilities. It highlights the integration of networking, database management, and GUI design to create a practical monitoring utility. Although it serves as a foundational tool, it can be expanded with more advanced detection methods, real-time graphs, and notification systems for enhanced security monitoring.