Network Packet Sniffer with Alert System

Introduction

In today's interconnected digital landscape, network security is paramount. With the rise of cyber threats such as unauthorized access, data breaches, and denial-of-service attacks, monitoring network traffic in real-time has become essential. This project focuses on developing a network packet sniffer equipped with an anomaly detection system. The tool captures and analyzes packets, detects suspicious activities like port scanning or flooding, logs data for forensic analysis, and alerts users when thresholds are breached. By integrating packet sniffing with alerting mechanisms, the system enhances proactive threat detection, making it valuable for network administrators and security enthusiasts. The objective is to build a functional CLI or GUI-based sniffer using open-source tools, ensuring it is efficient, user-friendly, and extensible.

Abstract

This report outlines the development of a real-time network traffic sniffer with anomaly detection capabilities. Utilizing Python as the core language, along with libraries like Scapy for packet manipulation, SQLite for data storage, and Matplotlib for visualization, the project creates a tool that captures packet headers, identifies anomalies such as port scans or packet floods, stores logs in a database, and triggers alerts via email or logging upon detecting breaches. An optional GUI component provides live traffic graphs for better insights. The system delivers a CLI/GUI packet sniffer with alerting and database logging, promoting enhanced network monitoring and security. The implementation emphasizes simplicity, efficiency, and real-time responsiveness, resulting in a practical solution for detecting and mitigating network threats.

Tools Used

The project leverages the following open-source tools and libraries:

- **Python**: Serves as the primary programming language for scripting the sniffer logic, due to its versatility and extensive ecosystem.
- **Scapy**: A powerful packet manipulation library used for capturing, forging, and dissecting network packets in real-time.
- **SQLite**: A lightweight, embedded database for storing captured packet data, enabling efficient querying and long-term logging without requiring a full database server.
- **Matplotlib**: Employed for generating visualizations, such as traffic summaries and live graphs in the optional GUI component.

These tools were selected for their compatibility, performance in handling network data, and ease of integration in a Python environment.

Steps Involved in Building the Project

The project was developed through a structured, iterative process:

1. **Packet Capture and Logging**: Using Scapy, the system captures network packets in real-time via a specified interface (e.g., Ethernet or Wi-Fi). Key headers such as IP addresses, ports, packet length, and flags (e.g., SYN, ACK) are extracted and logged. This forms the foundation for analysis, ensuring only relevant data is processed to avoid overhead.
2. **Anomaly Detection**: Algorithms were implemented to detect common anomalies. For port scanning, the system tracks connection attempts across multiple ports from a single IP, flagging rapid scans. Flooding is identified by monitoring packet rates exceeding predefined thresholds (e.g., >100 packets/second from one source). Custom rules based on packet patterns enhance detection accuracy.
3. **Data Storage and Summary Display**: Captured data is inserted into an SQLite database with tables for packets, anomalies, and summaries. Queries generate traffic overviews, such as top IPs or port usage, displayed via console output or Matplotlib charts for visual summaries.
4. **Alert System**: Thresholds for anomalies (e.g., scan attempts >5 in 10 seconds) trigger alerts. Integration with Python's smtplib sends email notifications, while logging uses the built-in logging module for file-based alerts. This ensures immediate awareness of potential threats.
5. **Optional GUI Integration**: A simple GUI was added this displays real-time packet flows, anomaly alerts, and database queries in an interactive dashboard, enhancing usability for non-technical users.

Conclusion

This network packet sniffer with an alert system successfully achieves real-time monitoring and anomaly detection, providing a robust tool for network security. By capturing packets, storing data efficiently, and alerting on threats, it addresses key challenges in cybersecurity. The use of Python and associated libraries ensures the solution is accessible and scalable. Overall, the project demonstrates the power of open-source tools in building effective security applications, contributing to safer digital environments.