Network Packet Sniffer with Alert System

Introduction

Cybersecurity threats are growing rapidly, making real-time network monitoring an essential defense mechanism. This project presents a network packet sniffer with an integrated alert system that identifies suspicious network behavior such as port scanning and flooding attacks. The tool captures, logs, analyzes, and visualizes TCP packet traffic in real-time, providing both database records and anomaly alerts.

Abstract

This Python-based system captures live network packets using scapy, logs them into an SQLite database, and continuously analyzes the traffic to detect patterns of network anomalies. When a threshold is breached (indicating potential malicious activity), the system logs an alert and optionally notifies the user. A matplotlib-powered graph displays live traffic trends. The project is designed for educational and security analysis use.

Tools Used

- Python 3 – Core programming language
- Scapy – For capturing and dissecting packets
- SQLite3 – Lightweight database for logging packet data
- Matplotlib – For plotting live traffic data
- Linux (Kali) – OS environment for development and testing
- Terminal/Nano – For file handling and editing

Steps Involved in Building the Project

1. Environment Setup: Installed Python packages and created the project structure with key files: sniffer.py, db_handler.py, anomaly_detector.py, alert.py, and plotter.py.
2. Packet Capture: Built a real-time packet sniffer using scapy to capture TCP packet headers (IP, port, flags, etc.).
3. Database Logging: Used SQLite to log packet data with timestamps for future analysis.
4. Anomaly Detection: Implemented a detection algorithm to monitor IPs that generate excessive traffic in a short time window (threshold-based logic).
5. Alert System: Wrote logs to a file whenever a suspicious IP triggered an alert.
6. Traffic Visualization: Created a dynamic plot with matplotlib to show the number of packets per second.

7. Testing: Deployed and tested the system in a Kali Linux environment using simulated traffic patterns.

Conclusion

The project successfully demonstrates how to build a lightweight, real-time packet monitoring system with anomaly detection. It provides hands-on understanding of network behavior analysis, security monitoring, and Python-based automation. The modular design ensures it can be extended with features like protocol filtering, email alerts, or GUI-based dashboards in the future.