Network Packet Sniffer with Alert Messages

Introduction :

Network security is crucial in today's digital world, where unauthorized access, data breaches, and other cyber threats create serious risks. Real-time traffic monitoring helps detect suspicious behavior and enables quick response. The Network Packet Sniffer with Alert System is a Python-based tool that captures, analyzes, and visualizes network traffic, detects anomalies, and sends email alerts. With a user-friendly GUI and real-time analytics, it strengthens network security by giving administrators actionable insights to reduce risks effectively.

Abstract :

This report presents a real-time network traffic monitoring system developed using Python and Scapy. It captures network packets, extracts essential details (IP addresses, ports, protocols), and stores them in an SQLite database for analysis. Anomaly detection uses predefined rules like unusual port activity or high traffic volume, triggering email alerts through SMTP with the email.mime.text module. The system includes a Tkinter GUI and a Dash dashboard with Plotly and Matplotlib for real-time visualization. Multithreading supports smooth packet capture and processing. The result is improved network visibility, automated threat detection, and interactive insights for effective security management.

Tools Used :

Python: Core programming language for scripting and logic implementation.

Scapy: For packet capturing and analysis, enabling real-time sniffing of network interfaces.

email.mime.text: To construct and send alert emails via SMTP.

Dash and Plotly: For creating interactive web-based dashboards and visualizations.

Pandas: For data manipulation and analysis of captured traffic data.

SQLite3: Lightweight database for storing packet details (e.g., IP addresses, ports, protocols).

Tkinter: For building a desktop GUI with live-tracking features.

Threading: To handle concurrent tasks like packet capture, analysis, and GUI updates without blocking.

Matplotlib: For generating static and dynamic plots of traffic statistics.

Steps Involved in Building the Project :

- **Environment Setup:**
  - Installed required libraries (scapy, pandas, matplotlib, dash, plotly, tkinter, etc.).
  - Configured the environment with administrator privileges for packet capture.
  - Set up an SQLite database to store packet information (source/destination IP, port, protocol).
- **Packet Capture and Analysis:**
  - Used Scapy to capture network packets in real-time.
  - Parsed packets to extract relevant fields (source/destination IP, port, protocol).

- Implemented logic to detect anomalies (high packet rates, blacklisted IPs) using predefined rules.

➢ **Data Storage:**
- Stored captured packet data in an SQLite database for persistence and querying.
- Utilized Pandas for efficient data processing and filtering.

➢ **Alert System:**
- Configured the email.mime.text module to send email alerts when anomalies are detected.
- Integrated anomaly detection logic to trigger alerts based on thresholds or patterns.

➢ **Visualization and GUI:**
- Developed a Tkinter-based GUI to display live packet information and system status.
- Used Matplotlib for static graphs (e.g., traffic volume over time).
- Built an interactive dashboard with Dash and Plotly for real-time traffic visualization.

➢ **Multithreading:**
- Implemented threading to handle simultaneous packet capture, analysis, and GUI updates without performance bottlenecks.

➢ **Testing and Validation:**
- Tested the system on a local network to verify packet capture accuracy and anomaly detection.
- Validated email alerts and ensured visualizations accurately reflected traffic patterns.
- Optimized performance to handle high-traffic scenarios.

Conclusion :

The Network Packet Sniffer with Alert System is a powerful real-time monitoring and threat detection tool. Using Python, Scapy, and SQLite, it captures and stores packet data, while Pandas and visualization tools (Matplotlib, Dash, Plotly) offer clear traffic insights. A Tkinter GUI and multithreading improve usability and performance, and an email alert system ensures quick response to suspicious activity. This project integrates multiple technologies into a scalable, user-friendly network security solution, with future potential for machine learning-based anomaly detection.