Technical Report: Network Packet Sniffer with Alert System

1. Introduction

This project focuses on building a real-time network packet sniffer with an integrated alert system for detecting suspicious traffic activity. The goal is to monitor network packets, log relevant information into a database, detect anomalies like flooding or DoS attempts, and issue alerts through emails or logs. A graphical interface is also provided to visualize traffic over time.

2. Tools & Technologies

- Programming Language: Python 3.10
- Libraries Used:
  - scapy – for packet sniffing
  - sqlite3 – for local database storage
  - matplotlib – for graph visualization
  - smtplib and email – for sending alert emails
- Interface: Command-line (CLI) & optional GUI

3. Implementation Details

- Packet Capturing:
  The main script (main.py) uses scapy to sniff incoming and outgoing packets in real-time. It captures source IP, destination IP, ports, timestamp, and protocol.
- Database Logging:
  All packet information is logged into a SQLite database (traffic_log.db) for storage and future analysis.
- Anomaly Detection:
  A function in alert.py monitors packet traffic from each source IP. If more than 10 packets are received within 10 seconds from a single IP, it is flagged as suspicious.
- Alert Mechanism:
  Alerts are:
  - Printed in the terminal
  - Logged for auditing
  - Emailed to a specified recipient using Gmail SMTP (if credentials are valid and correct)

- Traffic Simulation:
  The script test_requests.py sends multiple HTTP requests to simulate high traffic and test the sniffer and alert system.
- Graphical Visualization:
  The gui_graph.py file reads from the database and plots traffic trends over time using matplotlib. This helps in identifying spikes in traffic visually.

4. Outputs & Observations

- Terminal Alerts: Suspicious IPs generate warning logs like:
  "[ALERT] High traffic from 192.168.1.5 (15 packets in 10s)"
- Email Alerts: Alerts are sent to your email using smtplib (once valid credentials are provided and Gmail settings are properly configured).
- Database View: Using view_logs.py, you can display the last 10 captured packets for review.
- Graph View: The GUI shows a live traffic graph with the number of packets over time.

5. Conclusion

This project demonstrates how packet sniffing and real-time alerting can be implemented using Python. It is effective for monitoring DoS attacks, port scanning, or suspicious network spikes. The system can be extended to detect more patterns, add firewall integration, or notify via other methods (like SMS or dashboards).