Network packet sniffer with alert system

1. Introduction

With the rapid growth of computer networks, monitoring and securing network traffic has become a crucial aspect of cybersecurity. Network sniffers are tools used to capture, log, and analyze network packets for both troubleshooting and security monitoring purposes.

This project focuses on building a **Python-based Network Sniffer** that can capture packets, store them in an SQLite database, detect suspicious activities like port scans, and visualize network behavior.

---

2. Objectives

- To capture live network traffic (TCP, UDP, ICMP).
- To store packet details in a structured database (SQLite).
- To implement **alert mechanisms** for detecting suspicious activity.
- To simulate attacks (port scanning) for testing IDS capabilities.
- To visualize traffic statistics using graphs.

---

3. System Requirements

Hardware

- Processor: Intel i3 or above
- RAM: 4 GB minimum
- Storage: 500 MB free space

Software

- Python 3.8+
- Required Libraries: Scapy, Matplotlib, Pandas
- SQLite (inbuilt with Python)
- Operating System: Windows / Linux

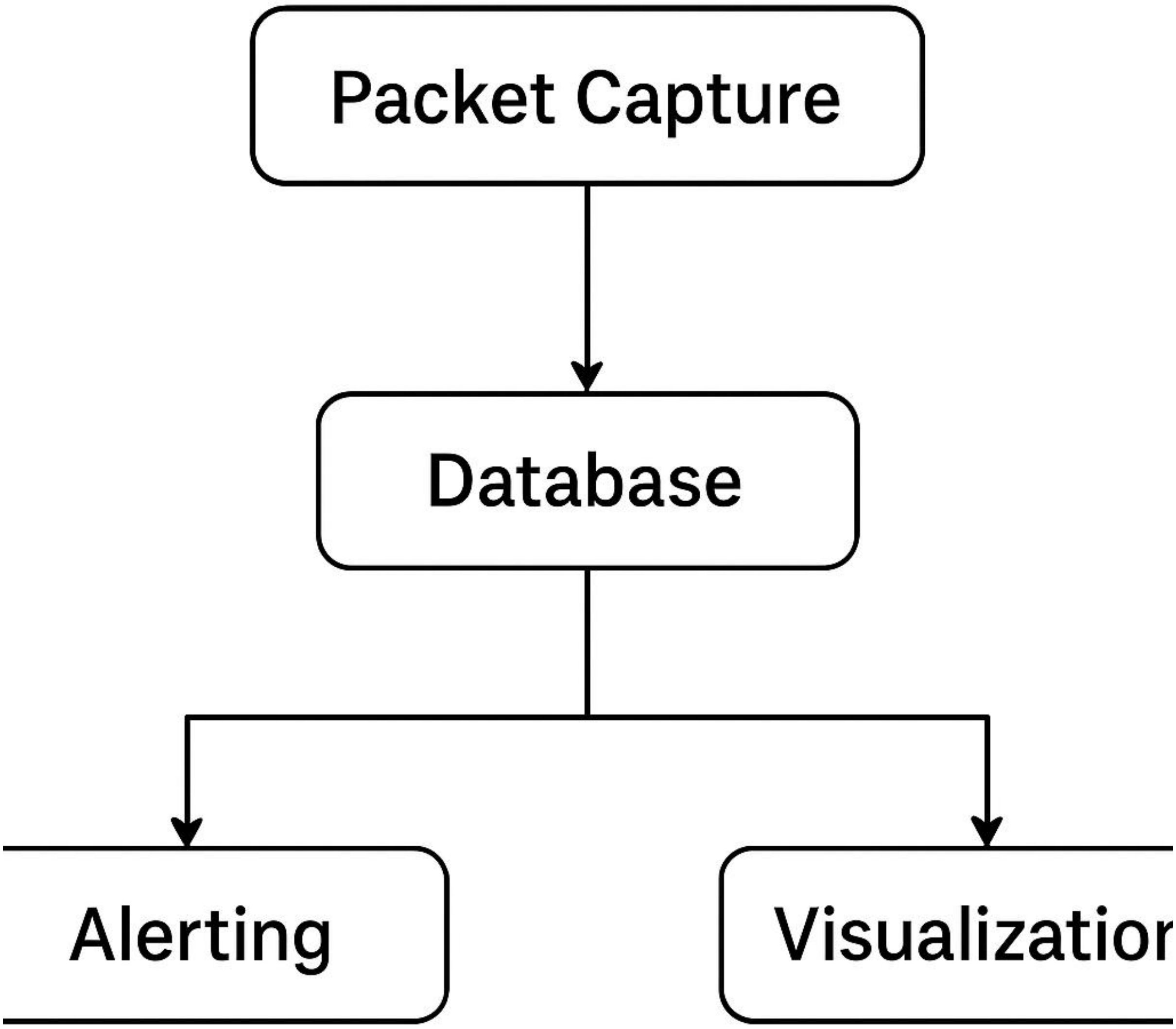
---

4. System Design

4.1 Architecture

1. **Packet Capture Layer** – Uses Scapy to sniff network traffic.
2. **Database Layer** – Stores packet information in packets.db.
3. **Alerting Module** – Detects suspicious patterns (port scans, abnormal traffic).
4. **Traffic Visualization** – Generates graphical insights.

4.2 Flow Diagram



5. Modules

1. **database.py** – Creates and manages SQLite database.
2. **sniffer.py** – Captures network traffic and inserts into database.
3. **packets.py** – Defines packet schema and database operations.
4. **alert.py** – Scans stored packets and raises alerts for abnormal traffic.
5. **testportscan.py** – Simulates port scanning for testing.
6. **udp.py** – Captures only UDP packets.
7. **visualization.py** – Creates graphs and charts for traffic analysis.

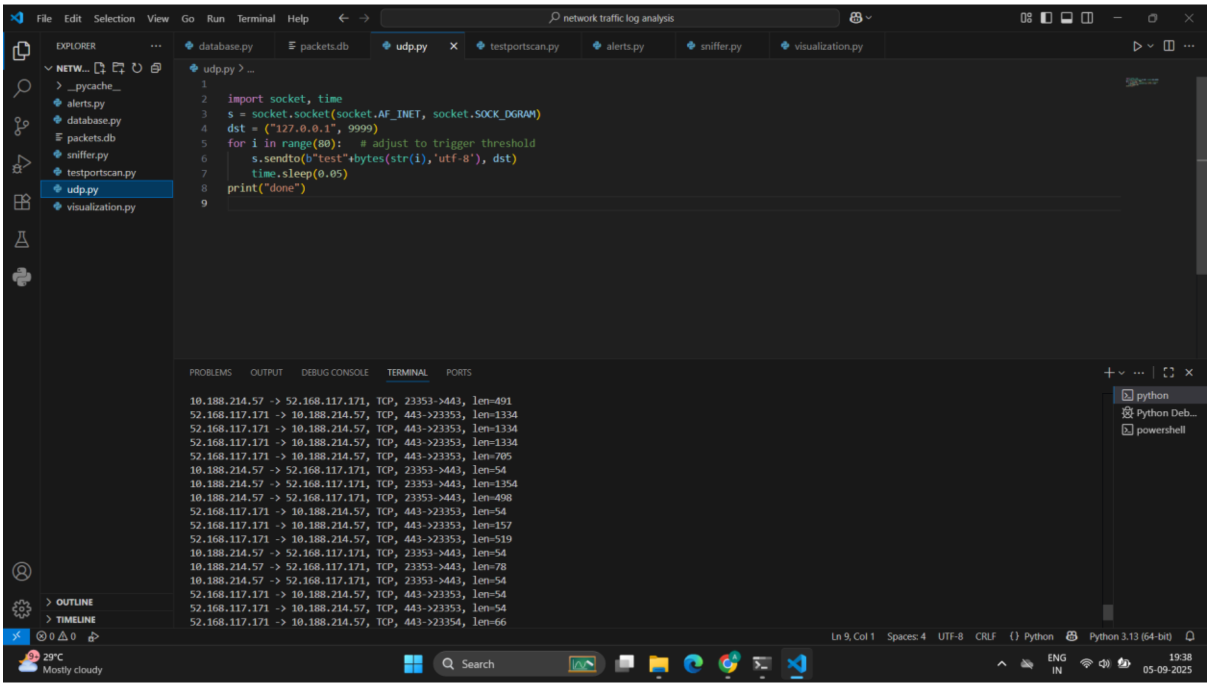
---

6. Implementation Steps

1. **Database Initialization**
   - Run database.py to create packets.db.
2. **Start Sniffer**
   - Run sniffer.py with admin/root privileges.
3. **Generate Traffic**
   - Use testportscan.py to simulate scanning attacks.
4. **Trigger Alerts**
   - Run alert.py to detect malicious activity.
5. **Visualization**
   - Run visualization.py to generate traffic analysis graphs.

---

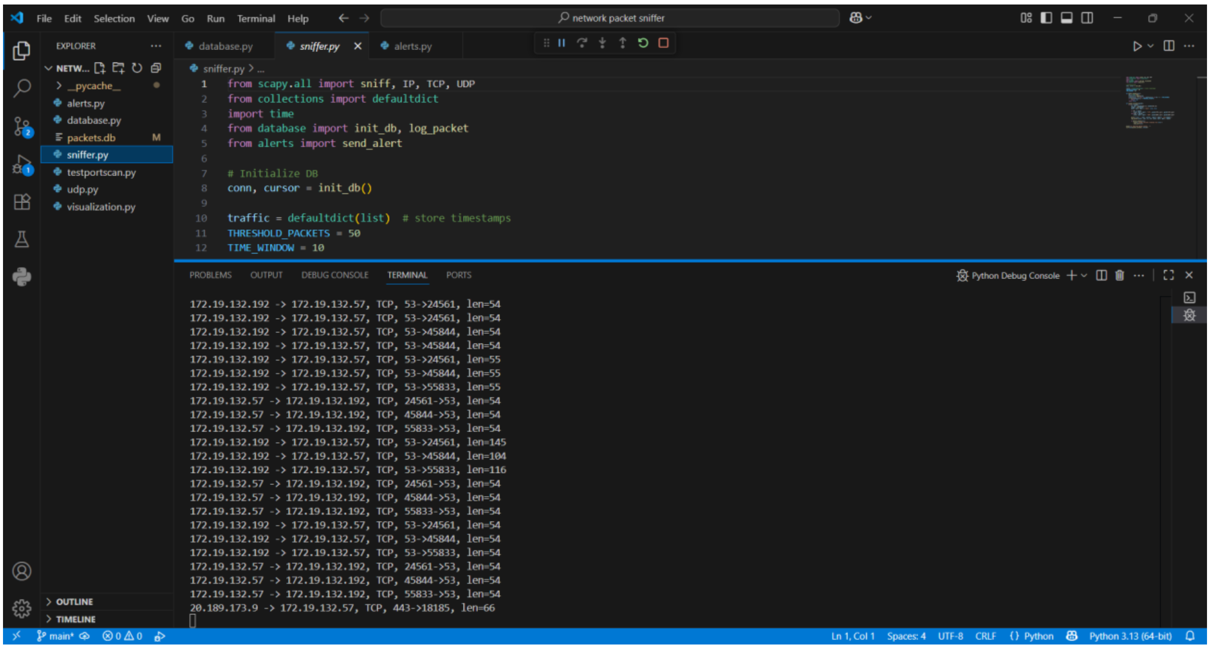
7. Sample Output



```python

import socket, time
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
dst = ("127.0.0.1", 9999)
for i in range(80):   # adjust to trigger threshold
    s.sendto(b"test"+bytes(str(i),'utf-8'), dst)
    time.sleep(0.05)
print("done")
```

```
10.188.214.57 -> 52.168.117.171, TCP, 23353->443, len=491
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=1334
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=1334
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=1334
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=705
10.188.214.57 -> 52.168.117.171, TCP, 23353->443, len=54
10.188.214.57 -> 52.168.117.171, TCP, 23353->443, len=1354
10.188.214.57 -> 52.168.117.171, TCP, 23353->443, len=498
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=54
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=157
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=519
10.188.214.57 -> 52.168.117.171, TCP, 23353->443, len=54
10.188.214.57 -> 52.168.117.171, TCP, 23353->443, len=78
10.188.214.57 -> 52.168.117.171, TCP, 23353->443, len=54
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=54
52.168.117.171 -> 10.188.214.57, TCP, 443->23353, len=54
52.168.117.171 -> 10.188.214.57, TCP, 443->23354, len=66
```

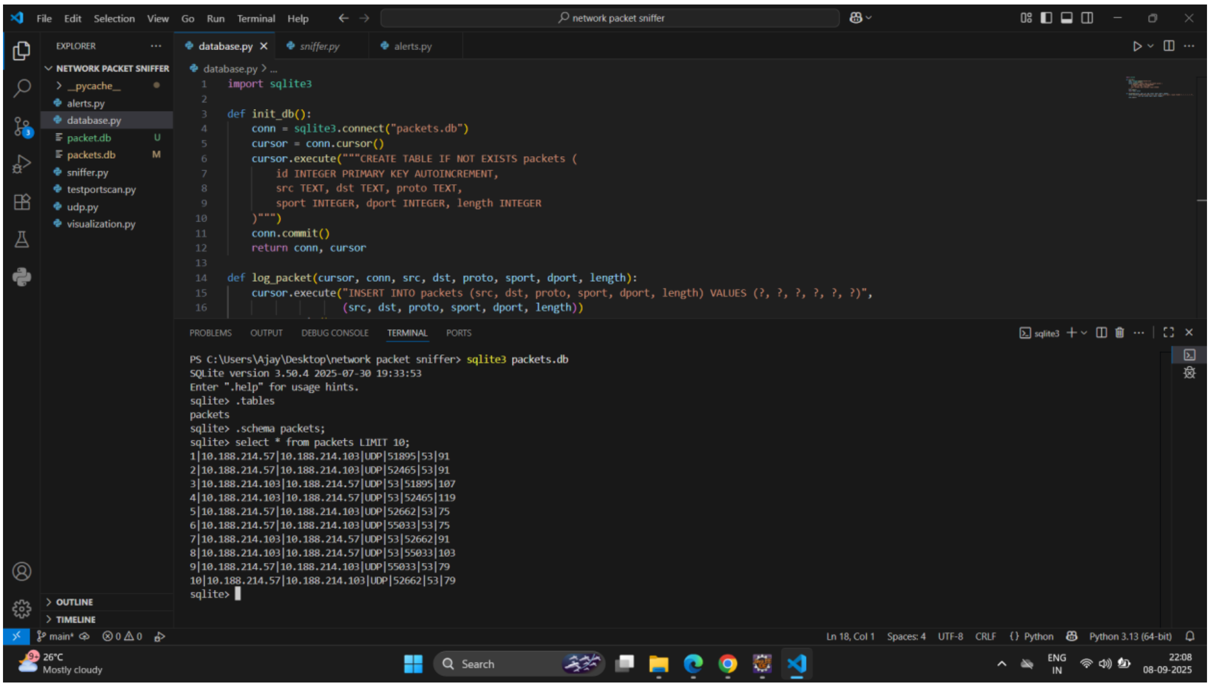


```python
from scapy.all import sniff, IP, TCP, UDP
from collections import defaultdict
import time
from database import init_db, log_packet
from alerts import send_alert

# Initialize DB
conn, cursor = init_db()

traffic = defaultdict(list)  # store timestamps
THRESHOLD_PACKETS = 50
TIME_WINDOW = 10
```

```
172.19.132.192 -> 172.19.132.57, TCP, 53->24561, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->24561, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->45844, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->45844, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->24561, len=55
172.19.132.192 -> 172.19.132.57, TCP, 53->45844, len=55
172.19.132.192 -> 172.19.132.57, TCP, 53->55833, len=55
172.19.132.57 -> 172.19.132.192, TCP, 24561->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 45844->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 55833->53, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->24561, len=145
172.19.132.192 -> 172.19.132.57, TCP, 53->45844, len=104
172.19.132.192 -> 172.19.132.57, TCP, 53->55833, len=116
172.19.132.57 -> 172.19.132.192, TCP, 24561->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 45844->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 55833->53, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->24561, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->45844, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->55833, len=54
172.19.132.57 -> 172.19.132.192, TCP, 24561->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 45844->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 55833->53, len=54
20.189.173.9 -> 172.19.132.57, TCP, 443->18185, len=66
```

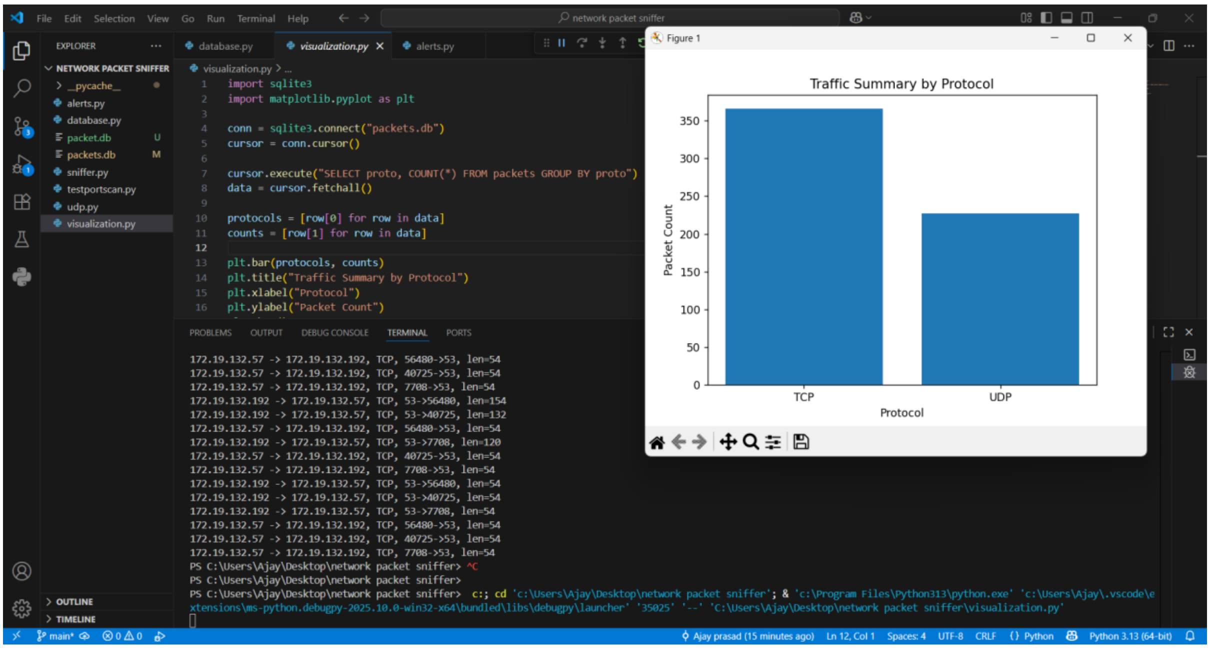


```python
import sqlite3

def init_db():
    conn = sqlite3.connect("packets.db")
    cursor = conn.cursor()
    cursor.execute("""CREATE TABLE IF NOT EXISTS packets (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        src TEXT, dst TEXT, proto TEXT,
        sport INTEGER, dport INTEGER, length INTEGER
    )""")
    conn.commit()
    return conn, cursor

def log_packet(cursor, conn, src, dst, proto, sport, dport, length):
    cursor.execute("INSERT INTO packets (src, dst, proto, sport, dport, length) VALUES (?, ?, ?, ?, ?, ?)",
                   (src, dst, proto, sport, dport, length))
```

```
PS C:\Users\Ajay\Desktop\network packet sniffer> sqlite3 packets.db
SQLite version 3.50.4 2025-07-30 19:33:53
Enter ".help" for usage hints.
sqlite> .tables
packets
sqlite> .schema packets;
sqlite> select * from packets LIMIT 10;
1|10.188.214.57|10.188.214.103|UDP|51895|53|91
2|10.188.214.57|10.188.214.103|UDP|52465|53|91
3|10.188.214.103|10.188.214.57|UDP|53|51895|107
4|10.188.214.103|10.188.214.57|UDP|53|52465|119
5|10.188.214.57|10.188.214.103|UDP|52662|53|75
6|10.188.214.57|10.188.214.103|UDP|55033|53|75
7|10.188.214.103|10.188.214.57|UDP|53|52662|91
8|10.188.214.103|10.188.214.57|UDP|53|55033|103
9|10.188.214.57|10.188.214.103|UDP|55033|53|79
10|10.188.214.57|10.188.214.103|UDP|52662|53|79
sqlite> 
```



```python
import sqlite3
import matplotlib.pyplot as plt

conn = sqlite3.connect("packets.db")
cursor = conn.cursor()

cursor.execute("SELECT proto, COUNT(*) FROM packets GROUP BY proto")
data = cursor.fetchall()

protocols = [row[0] for row in data]
counts = [row[1] for row in data]

plt.bar(protocols, counts)
plt.title("Traffic Summary by Protocol")
plt.xlabel("Protocol")
plt.ylabel("Packet Count")
```

```
172.19.132.57 -> 172.19.132.192, TCP, 56480->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 40725->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 7708->53, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->56480, len=154
172.19.132.192 -> 172.19.132.57, TCP, 53->40725, len=132
172.19.132.57 -> 172.19.132.192, TCP, 56480->53, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->7708, len=120
172.19.132.57 -> 172.19.132.192, TCP, 40725->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 7708->53, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->56480, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->40725, len=54
172.19.132.192 -> 172.19.132.57, TCP, 53->7708, len=54
172.19.132.57 -> 172.19.132.192, TCP, 56480->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 40725->53, len=54
172.19.132.57 -> 172.19.132.192, TCP, 7708->53, len=54
PS C:\Users\Ajay\Desktop\network packet sniffer> ^C
PS C:\Users\Ajay\Desktop\network packet sniffer>
PS C:\Users\Ajay\Desktop\network packet sniffer>  c:; cd 'c:\Users\Ajay\Desktop\network packet sniffer'; & 'c:\Program Files\Python313\python.exe' 'c:\Users\Ajay\.vscode\e
xtensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '35025' '--' 'C:\Users\Ajay\Desktop\network packet sniffer\visualization.py'
```

8. Results

- Successfully captured and logged packets in SQLite database.
- Detected port scanning attempts using alert mechanism.
- Visualized traffic patterns such as protocol usage and packet counts.

---

9. Applications

- Network monitoring for administrators.
- Intrusion detection in small/medium networks.
- Educational tool for cybersecurity training.

---

10. Future Enhancements

- Integration with **Splunk / ELK Stack** for enterprise monitoring.
- Machine learning-based anomaly detection.
- Real-time dashboard using Flask or Django.
- Cloud monitoring support (AWS, Azure, GCP).

---

11. Conclusion

This project successfully demonstrates the fundamentals of **network monitoring and security** using Python. By combining packet sniffing, database logging, alerting, and visualization, it provides a simplified version of an Intrusion Detection System (IDS).

It can be extended into a more robust SIEM (Security Information and Event Management) solution with advanced analytics and scalability.

---

12. References

- Roesch, M. (1999). *Snort - Lightweight Intrusion Detection for Networks*.
- Scapy Documentation: https://scapy.net
- SQLite Documentation: https://www.sqlite.org/
- Matplotlib Documentation: https://matplotlib.org/