Network Traffic Analysis, Visualization and Real-Time Monitoring Using Python

1. Introduction

Modern computer networks generate massive volumes of data every second. Analyzing this traffic is essential for understanding network behavior, optimizing performance, and detecting security threats. Network traffic analysis focuses on examining packet-level information such as protocols, IP addresses, timestamps, and packet sizes to extract meaningful insights.

This project implements a **complete network traffic analysis pipeline** using Python. It includes:

- Offline packet analysis using PCAP files
- Data extraction and storage in CSV format
- Protocol and bandwidth visualization
- Real-time network monitoring and provide a clean and interactive monitoring dashboard
- Rule-based suspicious activity detection (mini IDS)

The project bridges **networking fundamentals, data analysis, and cybersecurity concepts** using practical tools and real traffic data.

2. Tools and Technologies Used

Tool	Purpose
Wireshark	Capture live network traffic and generate PCAP files

PyShark	Parse PCAP files and capture live traffic programmatically
Python	Core programming language
Pandas	Data manipulation and analysis
Matplotlib	Visualization of protocols and bandwidth
Streamlit	Real-time web dashboard
OS / Threading / Asyncio	System-level handling and real-time processing

3. Project Architecture and Workflow

The project follows a modular and structured workflow:

1. Capture network traffic using Wireshark
2. Parse PCAP files using PyShark
3. Extract essential packet fields
4. Store processed data in CSV format
5. Perform protocol and bandwidth analysis
6. Visualize traffic patterns
7. Monitor traffic in real time (Display in dashboard)

8. Detect suspicious behavior using predefined rules

This separation improves **readability, scalability, and maintainability**.

4. Traffic Capture and PCAP Processing

4.1 Packet Capture

Network packets are captured using Wireshark and saved as PCAP files. These files contain raw packet-level data, including timestamps, IP addresses, protocols, and packet lengths.

4.2 Reading PCAP Files Using PyShark

PyShark is used to parse PCAP files efficiently using lazy loading. To improve performance:

- Packet storage is disabled (`keep_packets=False`)
- A maximum packet limit is enforced

4.3 Extracted Packet Fields

The following fields are extracted from each packet:

- Time (`sniff_time`)
- Source IP
- Destination IP
- Protocol (`highest_layer`)
- Packet length (bytes)

These fields are sufficient for traffic pattern analysis, bandwidth calculation, and anomaly detection.

4.4 CSV Storage

Extracted packet data is stored in `traffic.csv`, enabling:

- Reusability without reprocessing PCAP files
- Faster analysis using Pandas
- Easy visualization and reporting

5. Offline Traffic Analysis

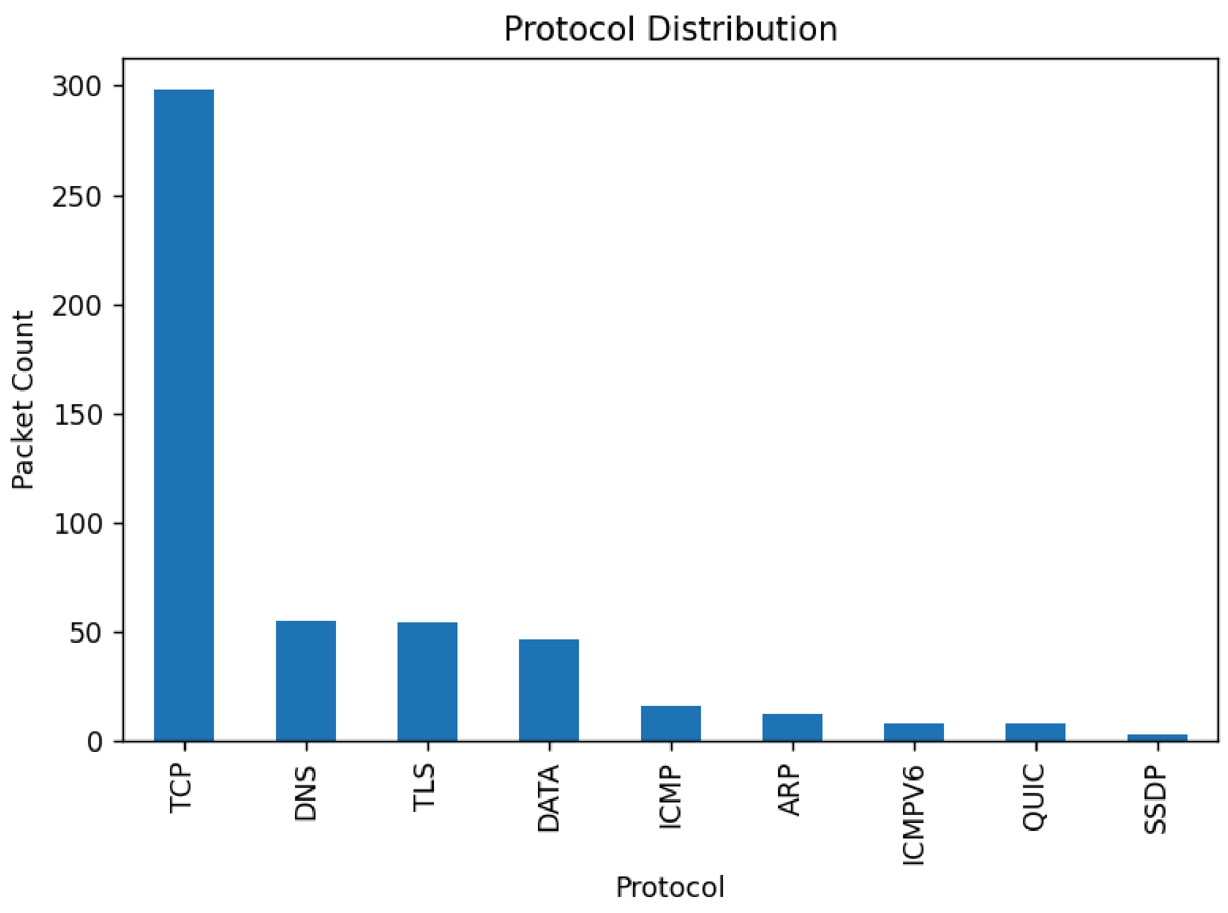
5.1 Protocol Distribution Analysis

Packet counts are grouped by protocol and visualized using bar charts.

Purpose:

- Identify dominant protocols (e.g., TCP, TLS)
- Detect unusual protocol usage
- Establish baseline network behavior

This analysis helps identify anomalies such as excessive ICMP or UDP traffic.

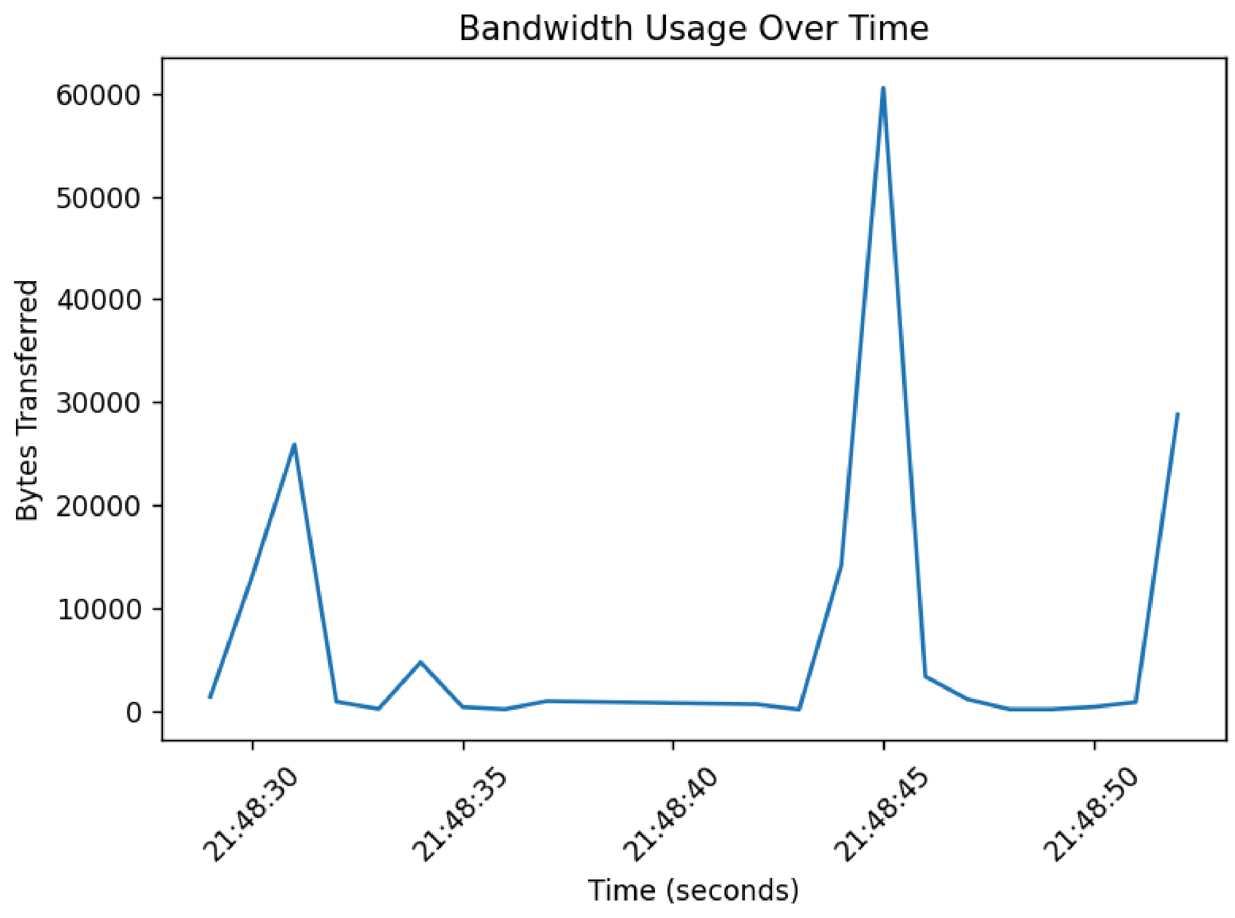


5.2 Bandwidth Usage Analysis

Bandwidth usage is calculated by:

- Converting packet timestamps to seconds
- Grouping packets per second
- Summing packet lengths

The resulting line graph shows **bytes transferred per second over time**.



Insights:

- Detect traffic spikes
- Identify congestion or heavy transfers
- Highlight abnormal bursts of traffic

5.3 Protocol-Level Bandwidth Analysis

Bandwidth usage is also analyzed per protocol, allowing comparison of how different protocols consume network resources. This helps detect:

- Protocol abuse
- Flood-based attacks
- Data-heavy encrypted traffic patterns

6. Data Cleaning and Validation

To ensure accurate analysis, the following preprocessing steps are applied:

- Conversion of packet length to numeric values
- Timestamp parsing and validation
- Removal of invalid or missing values
- Filtering valid IPv4 and IPv6 addresses
- Restricting analysis to known protocols

These steps improve **data reliability and visualization clarity**.

7. Visualization Strategy

Multiple visualizations are generated in a **single compact figure** for professional presentation:

- Packet count per protocol
- Bandwidth usage per protocol
- Top source IPs for selected protocols
- Protocol-wise bandwidth over time
- Highlighting suspicious protocols using color coding

This approach allows quick comparison and efficient interpretation.

8. Real-Time Network Monitoring

8.1 Live Packet Capture

Real-time traffic is captured directly from the network interface using PyShark's `LiveCapture`.

Key features:

- Thread-based packet capture
- Manual asyncio event loop handling (Python 3.12 compatibility)
- Sliding window storage for recent packets only

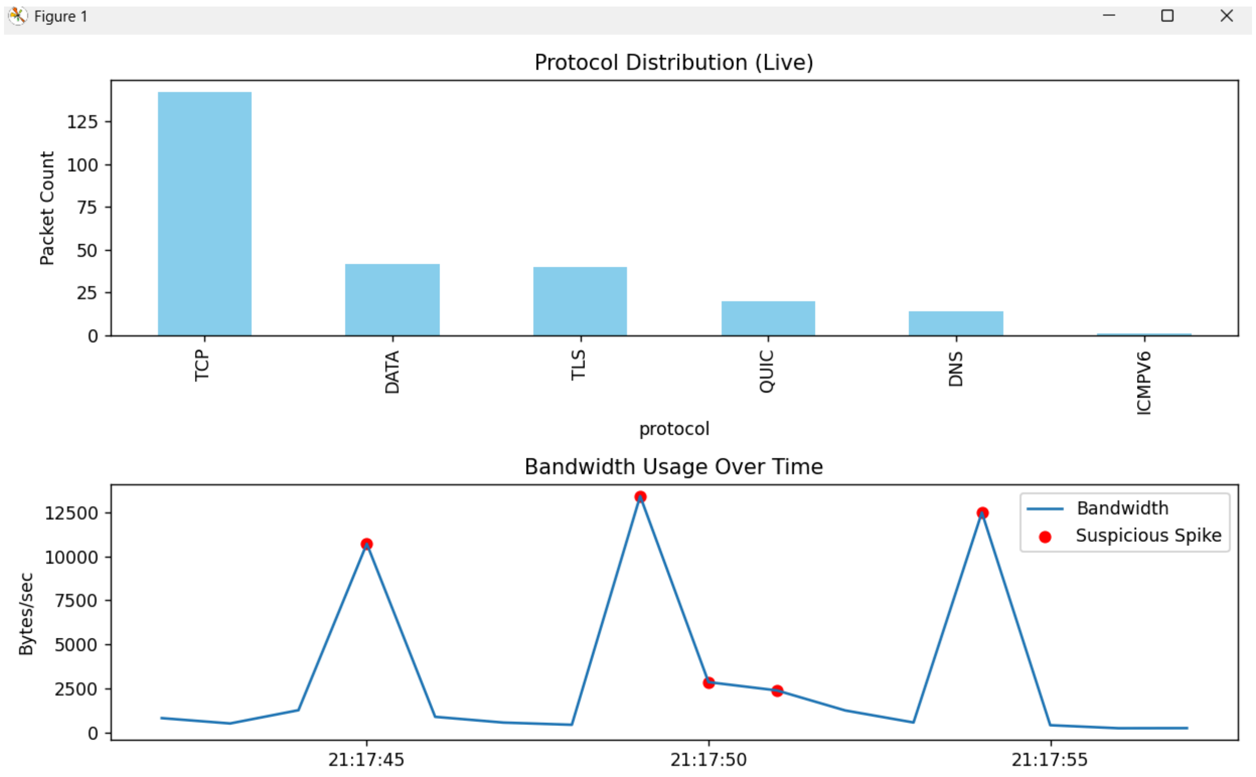
This ensures smooth, continuous monitoring without memory overflow.

8.2 Live Visualization

Two real-time plots are continuously updated:

- Protocol distribution
- Bandwidth usage over time

Bandwidth spikes are highlighted using red markers for immediate visibility.



9. Suspicious Activity Detection (Mini IDS)

9.1 Detection Rules

The system applies simple but effective rule-based detection:

Condition	Threshold
High packet rate from one IP	> 50 packets in 10 seconds
Bandwidth spike	> 2000 bytes/sec

9.2 Alerts and Visualization

- Suspicious IPs are printed as console alerts
- Bandwidth spikes are marked visually
- Abnormal protocol usage is highlighted in red

This mimics the core logic of real intrusion detection systems.

10. Protocol-Level Analysis

Traffic was analyzed separately for:

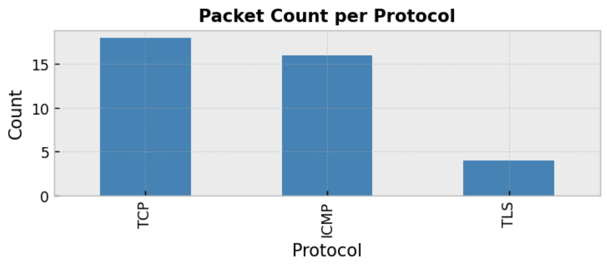
- TCP
- UDP
- TLS
- ICMP

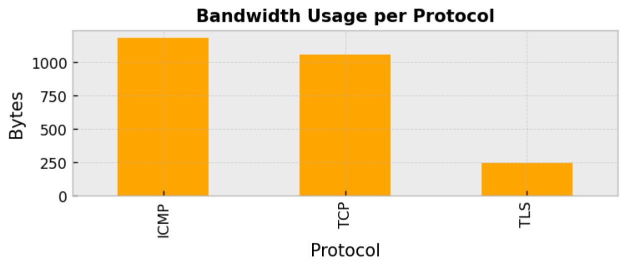
Benefits

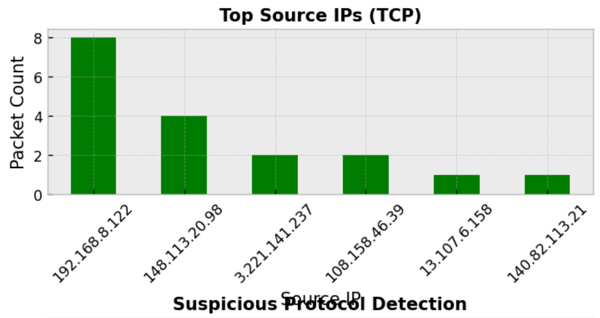
- Detect protocol misuse
- Identify flood attacks
- Understand protocol-specific behavior

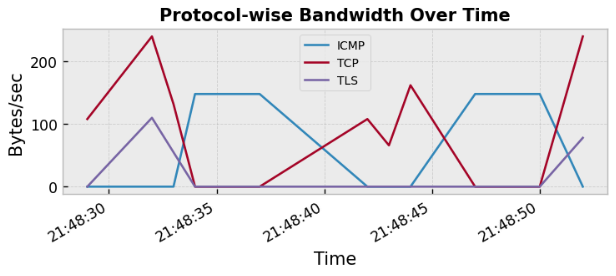
Example:

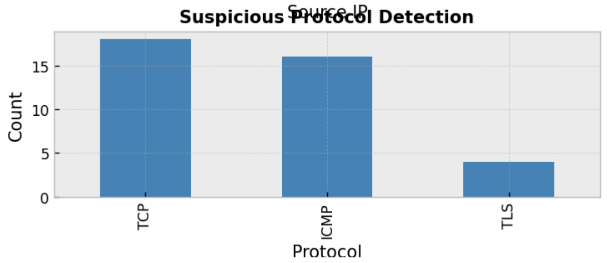
- Excessive ICMP → possible ping flood
- Sudden UDP surge → possible UDP flood


Packet Count per Protocol
Count
0
5
10
15
TCP
ICMP
TLS
Protocol


Bandwidth Usage per Protocol
Bytes
0
250
500
750
1000
ICMP
TCP
TLS
Protocol


Top Source IPs (TCP)
Packet Count
0
2
4
6
8
192.168.8.122
148.113.20.98
3.221.141.237
108.158.46.39
13.107.6.158
140.82.113.21
Source IP


Protocol-wise Bandwidth Over Time
Bytes/sec
0
100
200
ICMP
TCP
TLS
21:48:30
21:48:35
21:48:40
21:48:45
21:48:50
Time


Suspicious Protocol Detection
Count
0
5
10
15
TCP
ICMP
TLS
Protocol

11. Real-Time Network Traffic Analysis Dashboard

Objective

The objective of this dashboard is to monitor and analyze live network traffic in real time. It provides visual insights into protocol usage, bandwidth consumption, and potential security threats using packet-level data captured from the network.

System Overview

The dashboard was implemented using **Streamlit** for the web interface, **Pandas** for data processing, and **Matplotlib** for visualization. The system continuously reads live network traffic data stored in a CSV file and updates the dashboard automatically every five seconds.

Data Source and Handling

- Network packet data is stored in a continuously updated CSV file (`realtime_traffic.csv`).
- The dashboard safely reads the CSV file by checking:
    - Whether the file exists
    - Whether the file is empty
- If data is unavailable or being updated, the dashboard waits and refreshes automatically to avoid runtime errors.

This ensures uninterrupted real-time monitoring.

Data Cleaning and Preprocessing

Before visualization, the data undergoes preprocessing:

- Packet timestamps are converted to datetime format.
- Packet lengths are converted to numeric values.
- Invalid or missing entries are removed.
- Only the **most recent 30 seconds of traffic** are retained to maintain real-time relevance and reduce noise.

Protocol Filtering

A dynamic **protocol filter** is provided using a multiselect option. Users can:

- Select specific protocols (e.g., TCP, UDP, ICMP)
- View protocol-specific traffic statistics and graphs

This allows focused analysis of individual network protocols.

Protocol Distribution Visualization

- A bar chart displays the distribution of packets across different protocols.
- This helps identify dominant protocols and unusual traffic patterns.
- The graph uses a transparent background to visually integrate with the dashboard theme.

Bandwidth Usage Visualization

- Bandwidth usage is calculated by aggregating packet lengths per second.
- A time-series line graph shows bandwidth variation over the last 30 seconds.
- The time axis is formatted to display only `HH:MM:SS` and rotated to prevent overlapping labels.
- Transparent backgrounds and grid lines improve clarity and aesthetics.

This visualization helps detect traffic spikes and abnormal bandwidth consumption.

Suspicious IP Detection

- Source IP addresses are analyzed based on packet frequency.
- A threshold is defined to flag IPs generating an unusually high number of packets.
- If the threshold is exceeded, the IP address is marked as suspicious and displayed in a table.

This feature supports basic intrusion detection and anomaly monitoring.

Automatic Refresh Mechanism

- The dashboard refreshes automatically every five seconds.
- This is achieved using a controlled delay followed by a rerun command.
- Users receive near real-time updates without manual page refresh.

12. Performance Optimization

Optimizations include:

- Limiting packets stored in memory
- Processing packets on the fly
- Fixing asyncio conflicts in threaded environments

13. Limitations of the System

- No deep packet inspection
- Rule-based detection only
- Single-interface monitoring
- Limited scalability for very high-speed networks

14. Ethical and Legal Considerations

- Traffic capture is restricted to authorized environments
- Only metadata is analyzed
- Packet payloads are not inspected
- Intended for educational and research purposes

15. Future Enhancements

- Machine learning-based anomaly detection
- Real-time alert notifications
- IP geolocation analysis
- Containerized deployment
- Distributed traffic monitoring

---

16. Conclusion

This project successfully demonstrates a complete network traffic analysis and real-time monitoring system using Python. By integrating packet capture, data analysis, visualization, and basic intrusion detection, the system provides valuable insights into network behavior while remaining lightweight and educationally relevant.