Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 03, Issue 11, pp 68-72, November, 2026

REAL-TIME NETWORK BEHAVIOUR INTELLIGENCE

Ashaithambi M[1],Mukeshkumar R[2], Naga Arjun N[3], Surendiran M G[4], Vishnukumar S S[5]

[1]Assistant Proffesor
[2345] UG Scholars
[12345]Department of CSE (Cyber Security
[12345]Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India
Email : ashaithambi@siet.ac.in
Email : Mukeshkumarr23cys@srishakthi.ac.in
Email : nagaarjunn23cys@srishakthi.ac.in
Email : surendiranmg23cys@srishakthi.ac.in
Email : vishnukumarss23cys@srishakthi.ac.in

ABSTRACT :

Computer networks support communication between devices, applications, servers, and online services. Increasing network usage also creates a need for continuous monitoring and early identification of suspicious activities. This paper presents Real-Time Network Behaviour Intelligence, a Python-based network monitoring system that captures live packets using the Scapy library and analyzes their characteristics. The system extracts source and destination IP addresses, protocols, port numbers, packet sizes, and timestamps. It monitors traffic volume, connection frequency, and port activity to understand normal network behaviour and identify unusual patterns such as sudden traffic spikes, excessive connection attempts, and suspicious port usage. Detected abnormal behaviour can generate security alerts, while a web-based dashboard presents packet information, traffic patterns, anomalies, and alerts in an organized interface. The system combines packet capture, feature extraction, behavioural analysis, anomaly detection, and visualization to improve network visibility and support investigation of potential security issues. A detected anomaly indicates a potential risk and does not, by itself, confirm that an attack has occurred.

Keywords— Real-Time Network Monitoring, Network Behaviour Analysis, Scapy, Python, Packet Capture, Anomaly Detection, Network Security, Dashboard Visualization.

I. INTRODUCTION

Computer networks play an important role in connecting devices, applications, servers, and online services. As network usage continues to increase, monitoring network traffic and identifying security threats have become essential for protecting systems. Suspicious activities such as port scanning, excessive connection attempts, sudden traffic spikes, and unusual communication patterns can indicate potential security risks. Traditional monitoring approaches often focus on observing traffic and identifying known threats, which can make unusual behavioural changes difficult to detect quickly.

The Real-Time Network Behaviour Intelligence system addresses this need through continuous network traffic monitoring and analysis. It captures live network packets using Python and Scapy and extracts information including source IP addresses, destination IP addresses, protocols, ports, packet sizes, and timestamps. The collected information is examined to understand network connections and traffic volume. When suspicious traffic patterns or abnormal connection behaviour are identified, the system generates security alerts and presents network activity through a user-friendly dashboard.

II. PROBLEM STATEMENT

Network security is essential for protecting devices and data from potential cyber threats. However, large volumes of network traffic make manual monitoring difficult and time-consuming. Small networks may not have simple tools for identifying unusual traffic and potential security issues, while beginners may find it difficult to recognize sudden traffic spikes, excessive connections, or suspicious port activity. Traditional monitoring methods may also require technical knowledge and manual examination of packet data. The project therefore aims to develop a Python-based monitoring system using Scapy to capture live network packets, analyze traffic behaviour, identify unusual activities, generate real-time alerts, and display the results through a user-friendly dashboard.

III. OBJECTIVES

- To monitor network traffic continuously and observe network communication.
- To capture live network packets using Python and Scapy.
- To extract source IP, destination IP, protocol, port, packet size, and timestamp information.
- To identify normal network behaviour by analyzing traffic patterns and connections.
- To detect unusual traffic patterns, sudden traffic spikes, excessive connections, and suspicious port usage.
- To generate real-time security alerts whenever abnormal activities are detected.
- To display network traffic, detected anomalies, and alerts through a user-friendly dashboard.
- To help administrators understand network activities and investigate potential security issues.

IV. RELATED WORK

Network security research and practice include network traffic monitoring, behavioural analysis, anomaly detection, and intrusion detection techniques. Traditional monitoring tools allow administrators to observe network communication and inspect packet information. Wireshark is used to capture and examine individual packets; Zeek analyzes network traffic and produces detailed logs about connections and activities; and Suricata uses rules and signatures to identify suspicious traffic and potential attacks. Python-based packet-capture libraries such as Scapy provide programmable methods for collecting and analyzing packets.

Rule- and signature-based techniques are useful for recognizing known patterns, while behavioural monitoring examines changes in traffic volume, connection frequency, and port usage. The proposed project combines packet capture, traffic monitoring, behavioural analysis, anomaly detection, alert generation, and dashboard visualization in a single monitoring workflow. The project is intended as a practical monitoring aid and does not claim that every suspicious pattern is a confirmed attack.

V. PROPOSED SYSTEM ARCHITECTURE

The Real-Time Network Behaviour Intelligence system follows a modular architecture. The main stages are live network traffic capture, packet feature extraction, traffic monitoring and behavioural analysis, anomaly detection and alert generation, and dashboard visualization. Each stage processes information produced by the previous stage and contributes to the overall monitoring process.

A. Network Traffic Capture

The traffic capture module uses Python and the Scapy library to capture live network packets transmitted between connected devices. It collects packet information required for further analysis, including source and destination IP addresses, protocols, ports, packet sizes, and timestamps. Continuous packet capture provides the system with current information about network communication.

B. Packet Feature Extraction

The packet analysis module processes captured packets and extracts useful features. These include source IP address, destination IP address, protocol, port number, packet size, and timestamp. The extracted information is organized for traffic monitoring and behavioural analysis. Examining these features helps the system understand communication patterns and observe changes in network activity.

C. Traffic Monitoring and Behavioural Analysis

This module continuously observes network connections and traffic volume. It examines connection frequency, packet size, protocol usage, and port activity to understand normal network behaviour. Changes such as sudden increases in traffic, repeated connection attempts, and unusual port usage are treated as patterns that may require further investigation.

D. Anomaly Detection and Alert Generation

The anomaly detection module examines monitored traffic for unusual activities that may indicate potential security threats. Examples include sudden traffic spikes, excessive connections, and suspicious port usage. When abnormal behaviour is detected by the implemented detection logic, the system generates alerts to notify users. Alerts support investigation; they should not be interpreted as independent proof that a cyberattack has occurred.

E. Dashboard Visualization

The dashboard visualization module presents network traffic information, detected anomalies, and security alerts through a simple web-based interface. It can display packet details, traffic volume, protocol distribution, frequently accessed destination IP addresses, and recent alerts. Organizing this information in one interface helps users monitor network activities and understand changes in traffic behaviour.

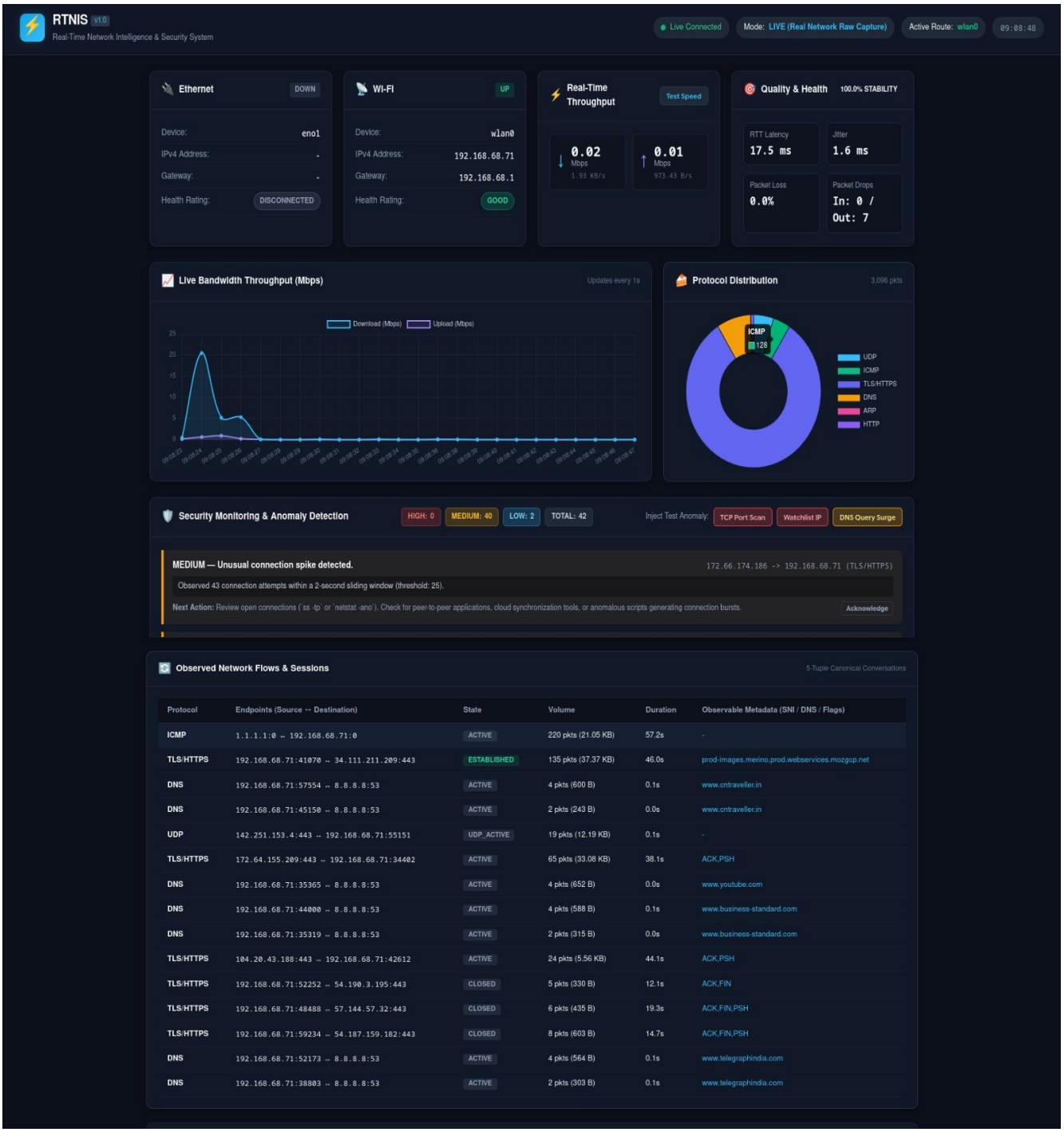
VI. WORKFLOW AND IMPLEMENTATION

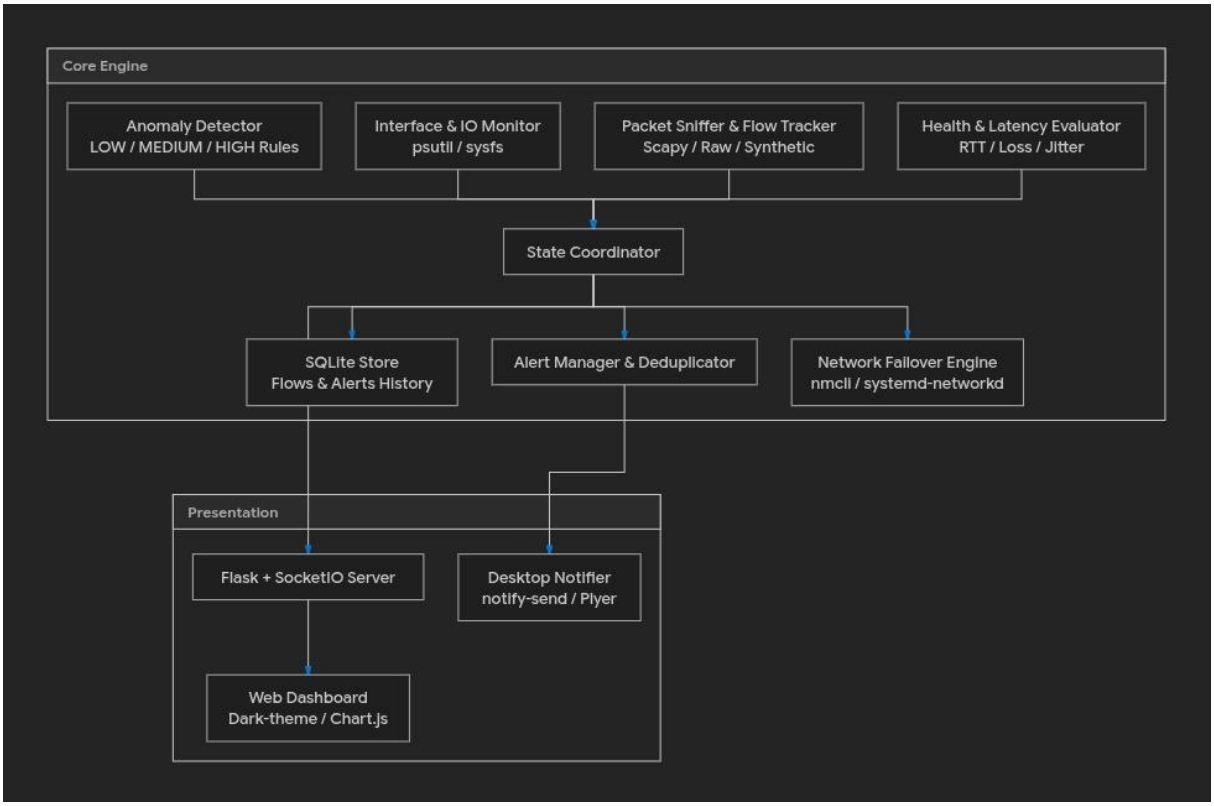
The workflow begins when the system captures live network packets using Python and Scapy. It then extracts packet details such as source IP, destination IP, protocol, port number, packet size, and timestamp. The extracted data is processed to monitor network traffic and analyze behavioural patterns. The anomaly detection stage examines the monitored information for suspicious changes, including sudden traffic spikes and excessive connections. When abnormal behaviour is detected, the system generates alerts. Finally, the dashboard displays network activities, packet details, and detected anomalies to support real-time monitoring.

- Step 1: Capture live network packets using Python and Scapy.
- Step 2: Extract packet features, including IP addresses, protocol, port, packet size, and timestamp.
- Step 3: Monitor traffic volume, connection frequency, and network communication patterns.
- Step 4: Analyze behaviour and identify unusual traffic patterns or suspicious activities.
- Step 5: Generate security alerts and display network information through the dashboard.

The software environment described for the project includes Python as the programming language and Scapy as the packet-capture library. HTML, CSS, and JavaScript are identified for the dashboard interface, with Python used for backend processing and alert logic. Visual Studio Code can be used as the development environment. Windows 10/11 or Ubuntu Linux are listed as suitable operating systems. Wireshark may be used during testing to inspect packets and compare traffic observations.

OUTPUT SCREENS





VII. RESULTS AND DISCUSSION

The project report describes an integrated monitoring workflow in which captured network packets provide the input for packet analysis, traffic monitoring, behavioural analysis, anomaly detection, and dashboard visualization. The Network Traffic Capture Module collects source and destination IP addresses, protocols, port numbers, packet sizes, and timestamps. These details provide the basic information needed to understand communication between connected devices and form the basis for further traffic analysis.

The Packet Analysis and Data Processing Module organizes captured packet information for examination. Packet size, protocol usage, port numbers, and source and destination addresses help users understand how network traffic flows through the monitored environment. The Traffic Monitoring and Behavioural Analysis Module observes traffic volume and connection frequency to identify changes that may differ from normal behaviour. Sudden increases in traffic, repeated connection attempts, and unusual port usage are examples of activities that may require investigation.

The Anomaly Detection and Alert Generation Module highlights unusual activities and can generate alerts when the implemented detection logic identifies abnormal behaviour. The dashboard brings together packet information, traffic patterns, and alerts. The described dashboard includes total captured packets, source and destination IP addresses, graphical representations of traffic volume and protocol distribution, live packet details, and recent alerts with severity levels.

The source project report does not provide numerical benchmark measurements for detection accuracy, false-positive rate, throughput, or response time. Therefore, no quantitative performance claim is made in this paper. Suspicious behaviour should be reviewed with supporting packet details and network context because an anomaly indicates a potential risk rather than proving malicious activity.

VIII. LIMITATIONS AND FUTURE WORK

The effectiveness of behavioural monitoring depends on the quality and coverage of captured traffic and the detection logic used. Unusual activity may have legitimate causes, and alerts may require manual investigation. The project report describes detection of traffic spikes, excessive connections, and suspicious port usage, but it does not provide a measured evaluation of accuracy or false-alert rates. Future evaluation should use representative network traffic and documented test cases to assess detection behaviour.

Future improvements may include integrating machine-learning algorithms to learn normal network behaviour and identify more complex or previously unseen anomalies. Configurable detection thresholds, detailed alert descriptions, and severity-based classification could help users prioritize events. Historical traffic analysis, automated report generation, interactive dashboard visualizations, improved filtering, and support for larger network environments may also improve the system's usefulness.

IX. CONCLUSION

The Real-Time Network Behaviour Intelligence system provides an approach to monitoring network traffic and identifying potentially suspicious

activities through continuous packet capture and behavioural analysis. Using Python and Scapy, the system captures live packets and extracts information such as IP addresses, protocols, ports, packet sizes, and timestamps. It monitors traffic volume, connection frequency, and port activity to identify unusual patterns such as sudden traffic spikes, excessive connections, and suspicious port usage. Alerts notify users when abnormal behaviour is detected, while the dashboard presents network traffic and detected activities in an organized form. The project demonstrates how packet capture, behavioural analysis, anomaly detection, and visualization can be combined to improve network visibility and support investigation of potential security issues.

REFERENCES

[1] G. Combs et al., "Wireshark," Wireshark Network Protocol Analyzer. https://www.wireshark.org/

[2] Zeek Project, "Zeek Network Security Monitor Documentation." https://docs.zeek.org/

[3] Open Information Security Foundation, "Suricata Documentation." https://docs.suricata.io/

[4] Scapy Project, "Scapy Documentation." https://scapy.readthedocs.io/

[5] Python Software Foundation, "Python 3 Documentation." https://docs.python.org/3/

[6] Sri Shakthi Institute of Engineering and Technology, "Real Time Network Intelligence Behaviour," Project Phase I Report, 2026.