Archives available at journals.mriindia.com

International Journal on Advanced Computer Theory and Engineering

ISSN: 2319-2526

Volume 15 Issue 01, 2026

Network Traffic Analyzer: Real-Time Detection and Malicious Activity Identification

[1]Anand Tamrakar, [2]Jagriti Chaturvedi, [3]Mansi Poddar, [4]Shreya Kumari
[1,2,3,4]Dept. of Computer Science and Engineering, SSIPMT, Raipur,India
[1]a.tamrakar@ssipmt.com, [2]jagriti.chaturvedi@ssipmt.com, [3]mansi.poddar@ssipmt.com, [4]shreyakumari@ssipmt.com

Peer Review Information

Submission: 29 March 2026
Revision: 17 April 2026
Acceptance: 21 May 2026

Keywords

Intrusion Detection System, Real-Time Monitoring, Network Traffic Analysis, Scapy, Streamlit, Cybersecurity, Comma-Seperated Values, Packet Capture, Domain Name System, Synchronize, Transmission Control Protocol.

Abstract

The design and implementation of a real-time net- work traffic analyzer that uses heuristic-driven analysis and live packet inspection for identifying malicious traffic are discussed in this research paper. The Scapy framework is used in the Python coding of the system's backend to enable continuous packet capture and processing with minimum overhead. As a way to identify abnormal behaviors such TCP SYN flooding, port scanning, suspicious DNS queries, sensitive payload content, and malformed packets, network traffic is evaluated. While the raw information is stored in PCAP format for ease of offline forensic analysis, detected events as well as packet summaries are stored in structured CSV files.

These logs are periodically read by a different Streamlit- based dashboard that uses an automatically refreshed interface to visualize alerts, protocol statistics, and host activity in real time. The suggested solution is appropriate for cybersecurity educa- tion, experimental research, and small-scale network monitoring deployments because it places an emphasis on transparency, low resource consumption, and modular design. In a variety of simulated attack scenarios, experimental evaluation shows prompt detection and dependable system responsiveness.

Introduction

Malicious actors gain entry to a larger attack surface as network infrastructures continue to expand in size and com- plexity. Using methods like TCP SYN flooding, port scanning reconnaissance, DNS tunneling, and covert data exfiltration, modern assaults have grown more common and complex. System uptime, data confidentiality, and general operational stability can all be severely impacted by these assaults. As a result, in order to minimize any adverse effects, companies need detection systems that can promptly and accurately identify threats.

Even though there are several commercially available de- tection systems for intrusions, many enterprise-grade solutions have real-world limitations. These systems sometimes operate as "black boxes" that require substantial processing power, rely heavily on periodic signature alterations, or provide no insight into how detection decisions are made. Due to these drawbacks, they are less suitable for academic contexts, small organizations, and those searching for easily comprehensible security technologies.

The objective of this endeavor is to develop a transparent, lightweight, real-time network traffic analyzer that uses fore- seeable and straight-forward methods to identify questionable activities. The main goal is to provide an approachable sys- tem that enables users to view unprocessed network activ- ity,

comprehend detection logic, and evaluate alarms without depending on intricate or resource-intensive infrastructures. The suggested solution allows for quick detection and clear presentation of network events by fusing Scapy-based packet processing with an interactive graphical user interface.

1. Problem Statement

High resource consumption, low explainability, and com- plicated deployment requirements are frequently encountered issues with traditional intrusion detection systems. Because of this, small businesses and educational institutions often lack useful solutions for efficient network monitoring. There is an obvious requirement for a system that:

- Live network traffic monitoring with little efficiency overhead,
- uses clear and understandable guidelines to identify ques- tionable activity,
- provides information and warning in an understandable visual style for those without expertise,
- encourages repeatable testing for scholarly and scientific objectives.

2. Contributions

The main contributions of this work are summarized as follows:

- A continuously running packet capture engine built on Scapy and Python that can handle large amounts of information.
- A Python and Scapy-based packet capture system that runs constantly and can process massive volumes of data.
- A host-based risk rating method to find high-risk or persistent sources.
- A fully decoupled, auto-refreshing Streamlit dashboard for traffic visualization and alert reporting.
- A modular framework that supports future extension using additional rules or machine learning techniques.

System Overview

Traffic capture and visualization are implemented as sepa- rate elements in the system architecture, that are technically decoupled. By keeping visualization tasks from interfering with real-time packet processing, this division enhances sta- bility and scalability. Although the frontend is designed for regular updates and user interaction, the backend concentrates on high-frequency tasks like packet sniffing, protocol parsing, and anomaly detection.

Lightweight, append-only data formats, such as CSV and JSONL files, are used for interactions between the dashboard and backend. This asynchronous system of communication ensures continuous monitoring even in the event that the visualization feature is momentarily unavailable by enabling the dashboard to poll updated information without interfering with packet capture.

A. Overall Architecture

As shown in Fig. 1,this packet analyzer continuously mon- itors a selected networking interface while processing each packet it collects using a multi-stage pipeline. This proce- dure includes gathering information, protocol recognition, and heuristic-based anomaly evaluation. Packet summaries and alerts have been copied to separate CSV files, but raw packets are retained for PCAP exportation to enable offline analysis.

Methodology

This work's methodology is centered on using a lightweight, organized algorithmic approach to evaluate network traffic in real time. A sequential analytic pipeline that finds protocols, collects metadata, and assesses them using heuristics comes after real-time packet collection from the selected network interface. Each processing stage is designed to minimize computational load while preserving vital information required for accurate anomaly detection. By combining deterministic rules with continuous recording and visualization, the method ensures timely detection, transparent decision-making, and repeatable analysis suitable for both experimental and operational situations.

1. Packet Capture and Processing Pipeline

Each captured packet passes through a structured processing pipeline:

1) Protocol Identification: Determination of protocol layers such as IP, TCP, UDP, DNS, or Raw.

2) Metadata Extraction: Extraction of timestamps, source and destination addresses, ports, protocol identifiers, payload sizes, and textual summaries.

3) Detection Invocation: Heuristic detection modules are used for evaluating packets.

4) Logging: Storage of structured metadata in CSV files and buffering of raw packets for PCAP export.

2. Heuristic-Based Detection Techniques

- SYN Flood Detection: The system records incoming TCP SYN packets by source IP address within a sliding time frame in order to identify attempts

at SYN flooding. A significant severity warning is triggered when the number of packets above a predetermined threshold.

- Port Scan Detection: By keeping track of how many different destination ports a single source contacts in a brief period of time, port scanning activity may be recognized. Both fast and slow scanning patterns may be found using this method.

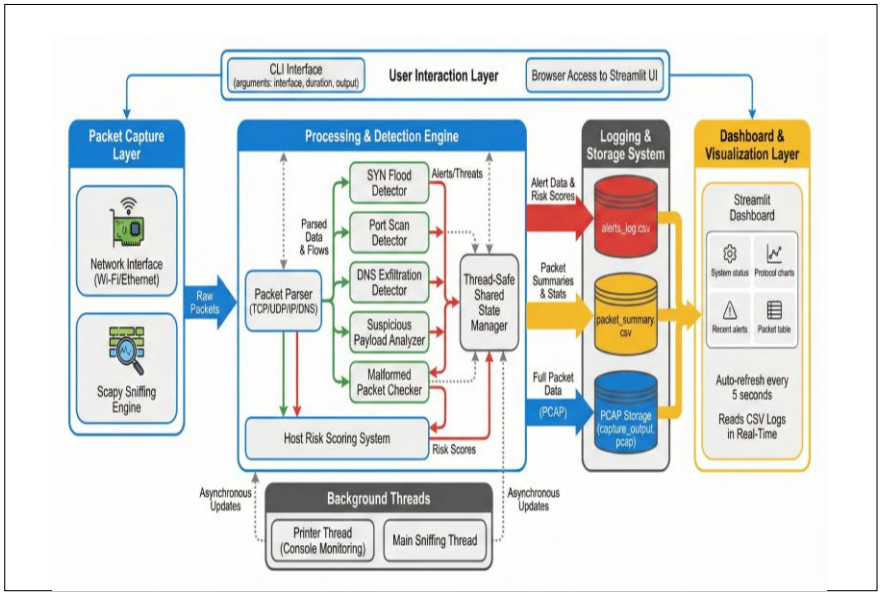

Fig. 1: System architecture for real-time packet analysis and dashboard visualization.

- DNS Exfiltration Detection: Unusually lengthy or strange QNAME fields in DNS requests are detected as possible signs of tunneling or secret data transmission.
- Suspicious Payload Detection: Packet payloads are in- spected for plaintext keywords such as “password”, “token”, or “cmd”, which may suggest credential exposure or command injection attempts.
- Malformed Packet Detection*:* Malformed packets are reported for additional examination when they show discrepan- cies between the claimed header lengths and the actual payload sizes.

Table 1: Detection Modules Implemented in the Proposed System

Module	**Functionality**
SYN flood detection	Tracks TCP SYN bursts per source IP
Port scan detection	Identifies multiple destination ports in short time window
DNS exfiltration detection	Flags oversized DNS QNAME queries
Suspicious payload detection	Matches plaintext security-sensitive keywords
Malformed packet detection	Detects inconsistencies in IP header fields
Host risk scoring	Aggregates traffic and alert severity per host

3. Host Risk Scoring

Each host is assigned a cumulative risk score that reflects its observed activity and alert severity:

$$Score = \alpha_1 \cdot Packets + \alpha_2 \cdot Bytes + \alpha_3 \cdot AlertSeverity$$

Dashboard Implementation

1. Status Computation

The dashboard summarizes system health using two states:

- ERROR-FREE — no alerts detected in the last 15 seconds.
- SUSPICIOUS — one or more recent alerts observed.

2. Auto-Refreshing User Interface

The user interface provides:

- Periodic automatic refresh,
- Dynamic alert and packet tables,
- Protocol distribution visualizations,
- Host-level traffic summaries.

Table 2: Detection Thresholds Used in the System

Detection Type	Threshold
SYN flood	≥ 50 SYN packets / 10 s
Port scan	≥ 30 destination ports / 10 s
DNS exfiltration	DNS QNAME > 100 bytes
Payload inspection	Keyword match occurrence
Malformed packets	Header size mismatch
Host risk score	Weighted alert accumulation

Experimental Results

1. Test Scenarios

The system was evaluated under simulated attack scenarios including accelerated SYN bursts, sequential and wide-range port scans, oversized DNS requests, and payloads containing suspicious keywords. These experiments were designed to verify the capability of the proposed system to detect different categories of malicious network behavior under realistic operating conditions.

Table 3: Experimental Performance Summary

Metric	Observed Value
Detection latency	1–5 s
Dashboard refresh interval	5 s
CPU utilization	< 15%
Memory usage	Low (metadata only)
Packet loss	None observed
False alert rate	Minimal

2. Performance Evaluation

Table 4 summarizes detection timing across different attack scenarios.

Table 4: Detection Behavior for Attack Scenarios

Scenario	Alert Type	Detection Time (s)
SYN Flood	SYN flood alert	2–3
Port Scan	Port scan alert	3–5
Long DNS Query	DNS exfil alert	1–2
Payload Keyword	Suspicious payload alert	1–2

3. System Responsiveness

The recommended system's responsiveness was evaluated using the time gap between the alert's production at the server side and its presentation on the Streamlit dashboard. This analysis was carried out in all sample scenarios to ensure consistent performance under different traffic conditions. The results showed that newly generated alerts consistently appeared within a single dashboard refresh cycle of about five seconds. This result confirms that the system architecture effectively supports asynchronous communication between the packet acquisition engine and the visual representation component.

The reactivity of the system is largely due to its decoupled architecture. The backend continuously logs packet summaries and alert data into lightweight, structured CSV and JSONL log files while the dashboard reads and updates the information it presents on its own. This segregation ensures that the computationally expensive tasks of packet sniffing and anomaly identification are not hampered by visualization processes.

As a result, especially in the midst of extremely high traffic volumes or erratic attack patterns, the system maintains steady operation. Notably, even during periods of increased network traffic and realistic attack bursts, the end-to-end delay between recognition and presentation was constant and within acceptable bounds. Network operators may promptly receive alerts and take the necessary corrective action due to its stability. All things considered, our findings demonstrate that the recommended approach is perfect for real-time network monitoring applications, particularly where effective situational consciousness and prompt incident response need low detect-to-display latency.

Conclusion

This study offers a modular and easily understood real- time network traffic analyzer that may be used in educational settings, research labs, and small-scale network monitoring in- stallations. The system combines Scapy-based packet capture with simple, heuristic-driven detection algorithms to provide comprehensive insight into network activities with minimal computational overhead. Unlike many conventional intrusion detection systems, the proposed analyzer prioritizes openness, allowing users to understand both the detected traffic patterns and the reasoning behind generated alarms.

The test results show that the system can reliably identify a range of harmful actions, including SYN flooding, port scanning, odd DNS queries, suspicious payload content, and malformed packets. Additionally, the use of a detached dash- board, which facilitates timely and informed decision-making without compromising packet processing efficiency, enables real-time viewing of alarms and traffic

statistics.

Even though the current design prioritizes clarity and sim- plicity, there are still some intriguing directions for future development. These include setting up scattered monitoring nodes over multiple network segments, using long-term traffic profiling for trend evaluation and forensic investigation, and combining statistical anomaly detection methods with machine learning models to identify subtle or previously undetected attack patterns. These improvements would improve the ac- curacy, scalability, and utility of the proposed network traffic analyzer.

References

J. McHugh, "Intrusion and intrusion detection," ACM Computing Sur- veys, vol. 33, no. 2, pp. 135–161, 2001.

M. Roesch, "Snort: Lightweight Intrusion Detection System for Net- works," in Proc. 13th USENIX Conference on System Administration (LISA), 1999.

G. Conti, Security Data Visualization: Graphical Techniques for Network Analysis. No Starch Press, 2007.

H. Shiravi, A. Shiravi, and A. A. Ghorbani, "Toward Systematic Visualization for Network Attack Detection," IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 3, pp. 318–331, 2012.

J. Liao, M. Sedlmair, M. A. Sedlmair, and T. Munzner, "Visual Analytics for Network Traffic," IEEE Security & Privacy, vol. 14, no. 4, pp. 62–70, 2016.

R. Bejtlich, The Tao of Network Security Monitoring. Addison-Wesley, 2004.

V. Paxson, "Bro: A System for Detecting Network Intruders in Real- Time," Computer Networks, vol. 31, no. 23–24, pp. 2435–2463, 1999.

M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Dataset," in Proc. IEEE CISDA, 2009.

N. Moustafa and J. Slay, "UNSW-NB15: A Comprehensive Dataset for Network Intrusion Detection Systems," IEEE Transactions on Network and Service Management, vol. 13, no. 1, pp. 1–15, 2016.

Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characteriza- tion," in Proc. ICISSP, 2018.

H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "LOF: Identifying Density-Based Local Outliers," in Proc. ACM SIGMOD, 2009, pp. 93–104.

S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521,

pp. 436–444, 2015.

Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

S. M. Kasongo and Y. Sun, "Performance Analysis of Intrusion Detec- tion Systems Using Feature Selection," Journal of Big Data, vol. 7, no. 1, 2020.

K. Demertzis, K. Tsiknas, C. Skianis, and L. Iliadis, "Darknet Traffic Big-Data Analysis and Real-Time Malicious Intent Detection," arXiv preprint arXiv:2102.08411, 2021.

Y. Singh and P. Sehgal, "Real-Time Packet Filtering Module for Network IDS," International Journal of Computer Trends and Technology, 2019.

V. Manne, Y. Reddy, S. Naik, and M. Charan, "Machine Learning-Based Deep Packet Inspection Framework," International Journal of Scientific Research in Science and Technology, 2022.

Sharma and M. Makwana, "Malicious Packet Detection Using Anomalous TTL Values," International Journal of Recent Innovations in Computer and Communication Engineering, 2023.

Sobrero, M. Clavarezza, D. Ucci, and I. Bisio, "Near Real-Time Protocol Tunneling Detection Based on Machine Learning," arXiv preprint arXiv:2309.12720, 2023.

M. Nakip and E. Gelenbe, "Online Self-Supervised Deep Learning for Intrusion Detection Systems," arXiv preprint arXiv:2306.13030, 2023.

T. Chen, Y. Zhang, and W. Lan, "RIDE: Real-Time Intrusion Detection via Explainable Machine Learning," arXiv preprint arXiv:2311.16018, 2023.

K. Kimanzi, R. Kimanga, and F. Cherori, "Deep

Learning Algo- rithms Used in Intrusion Detection Systems: A Review," arXiv preprint arXiv:2402.17020, 2024.

R. Ahmad et al., "Deep Learning Exploits in Network Intrusion Detec- tion Systems," IEEE Access, vol. 9, pp. 98036–98047, 2021.

Alothman, "Cybersecurity Dashboards for Security Operations Cen- ters," IEEE Access, vol. 11, pp. 60412–60427, 2023.

L. Zeng, Time-Series Visualization Techniques for Real-Time Monitor- ing. Springer, 2022.

J. Kreps, "The Log: What Every Engineer Should Know About Real- Time Data Pipelines," LinkedIn Engineering Blog, 2014.

Á. Miguel-Díez and C. Campazas-Vega et al., "A Systematic Review of Unsupervised Algorithms for Anomalous Traffic Detection," arXiv preprint arXiv:2503.08293, 2025.