Network Traffic Analyzer Using Wireshark/Scapy

Ruby Angel T G
Assistant Professor
Department of Information Technology
Sathyabama Institute of Science and Technology
Chennai, India
rubyangel.t.g.it@sathyabama.ac.in

Abinaya K
UG Student
Department of Information Technology
Sathyabama Institute of Science and Technology
Chennai, India
abinayamoorthy0379@gmail.com

Ranjini M
UG Student
Department of Information Technology,
Sathyabama Institute of Science and Technology
Chennai, India
ranjini11052006@gmail.com

Vinaya Sree K
UG Student
Department of Information Technology
Sathyabama Institute of Science and Technology
Chennai, India
vinayasree292@gmail.com

Keerthika Narayana Moorthy
UG Student
Department of Information Technology
Sathyabama Institute of Science and Technology
Chennai, India
shirs.keerthikanarayanan@gmail.com

Jayapriya R
UG Student
Department of Information Technology
Sathyabama Institute of Science and Technology
Chennai, India
jayapriya14092005@gmail.com

In modern computer networks, efficient monitoring and analysis of traffic are essential for ensuring security, performance, and reliability. This paper presents a comprehensive network traffic analyzer utilizing Wireshark and Scapy to capture, inspect, and interpret real-time packet data. Wireshark provides a graphical interface for detailed protocol analysis, while Scapy enables custom packet manipulation and automation through Python scripting. The proposed system identifies network anomalies, detects suspicious activities, and assists in troubleshooting communication issues. Experimental results demonstrate the tool's effectiveness in analyzing TCP/IP flows and detecting intrusions. This integration of open-source tools provides a flexible and cost-effective solution for network administrators and researchers. The study contributes to the field of network security and performance optimization through practical, scalable implementation.

I. Introduction

In the rapidly evolving landscape of digital communication, the efficiency and security of network systems have become critical to organizational success. With the exponential growth of connected devices, cloud-based applications, and data-driven services, the volume of network traffic has increased beyond traditional monitoring capabilities. Every packet traversing a network can carry essential insights—ranging from benign user activity to indicators of cyber threats such as data exfiltration or denial-of-service attacks. However, distinguishing between normal and malicious traffic amidst this overwhelming data flow poses a significant analytical challenge.[1]

Traditional network monitoring tools, while effective in static environments, often fall short when faced with dynamic and large-scale traffic patterns. This limitation has paved the way for intelligent and automated traffic analysis solutions that can capture, inspect, and classify packets in real time. In this context, open-source tools such as **Wireshark** and **Scapy** have emerged as powerful instruments for network researchers and security professionals. Wireshark provides a comprehensive, graphical interface for deep packet inspection, allowing visualization and protocol-level breakdown of network communications. Scapy, on the other hand, offers flexible Python-based packet crafting, sniffing, and scripting capabilities, enabling customized automation and analysis beyond the constraints of standard interfaces.[2]

The integration of these two tools forms the foundation of our proposed **Network Traffic Analyzer**, designed to enhance visibility into network behavior and improve incident response. By leveraging Scapy's programmable environment and Wireshark's analytical depth, the system provides both granular packet analysis and high-level traffic summaries. Furthermore, this framework assists in anomaly detection, intrusion analysis, and protocol performance assessment, making it suitable for academic research, enterprise diagnostics, and cybersecurity investigations.[3]

Ultimately, this work aims to bridge the gap between manual packet analysis and intelligent, automated network monitoring. The proposed system demonstrates how open-source technologies can be combined to deliver scalable, real-time network visibility without the prohibitive cost or complexity of commercial solutions. [4]

Recent increases in cyberattacks, data breaches, and bandwidth misuse underscore the urgent need for more adaptive network analysis tools. Many organizations struggle to detect and respond to attacks in time due to inefficient traffic visibility and slow forensic investigation processes. By integrating Wireshark's visualization strength with Scapy's automation capabilities, the proposed analyzer provides a balanced approach—offering both immediate insights for network administrators and programmable flexibility for research and experimentation.[5]

This paper is structured to provide a comprehensive exploration of the system's design and implementation. Section II reviews related works and existing traffic analysis tools. Section III details the architecture of the proposed system, including data capture, processing, and visualization layers. Section IV discusses the experimental setup, performance evaluation, and results obtained.[6]

II. LITERATURE REVIEW

[1] The exponential rise in digital communication has transformed how data traverses modern networks, creating an urgent need for advanced network traffic analysis techniques. As cyberattacks, data breaches, and performance bottlenecks continue to evolve, researchers have devoted extensive effort to developing tools and frameworks for packet inspection, anomaly detection, and network forensics. Network traffic analysis serves as the foundation for maintaining confidentiality, availability, and performance in both enterprise and academic environments, providing valuable insights into data flow behaviors and potential security threats.

[2] Early research in network monitoring primarily relied on basic tools such as **tcpdump** and **NetFlow**, which provided raw packet capture and basic flow statistics. While effective in smaller networks, these tools lacked the visualization and deep protocol analysis capabilities required for today's complex, multi-protocol architectures. Consequently, traditional network analyzers struggled to detect subtle anomalies or to process high-throughput data streams efficiently. This limitation paved the way for the development of advanced packet analyzers capable of decoding thousands of protocols in real time, providing a more holistic understanding of network behavior.

[3] **Wireshark**, one of the most prominent open-source network analyzers, has been extensively discussed in research for its powerful packet capture and inspection features. Studies such as those by Natarajan et al. [1] and Singh et al. [2] have demonstrated Wireshark's potential in identifying network attacks, debugging communication protocols, and improving network performance. With its intuitive graphical interface and deep packet inspection capabilities, Wireshark enables users to analyze live traffic and recorded traces with exceptional granularity, making it a cornerstone tool for cybersecurity education and research.

[4] Beyond graphical analysis, automation and programmability have emerged as key needs in modern traffic analysis. **Scapy**, a Python-based packet manipulation library, has gained popularity among researchers for its flexibility in packet crafting, sniffing, and analysis. Works by Bhatia and Gupta [3] and Rahman et al. [4] have highlighted Scapy's effectiveness in building customizable network testing tools and security simulators. Scapy's scripting capability allows analysts to automate repetitive network experiments, simulate attacks, and collect traffic data efficiently, thus bridging the gap between manual packet analysis and intelligent network automation.

[5] Recent studies have begun integrating Wireshark and Scapy to create hybrid frameworks for network monitoring. Combining Wireshark's visualization strengths with Scapy's programmable flexibility allows for both real-time inspection and automated anomaly detection. For example, Alam et al. [5] demonstrated a system that uses Scapy for automated packet generation while leveraging Wireshark for visual verification and deep analysis. Such hybrid frameworks empower researchers to monitor traffic dynamically and respond to network irregularities with both speed and accuracy. Integrating Wireshark and Scapy into hybrid frameworks enhances real-time monitoring, anomaly detection, and reporting. Such systems provide a scalable, flexible, and accessible solution for network research, performance optimization, and cybersecurity applications.

[6] In the realm of cybersecurity, network traffic analyzers have evolved to serve as frontline defense mechanisms. Researchers such as Li et al. [6] and Mishra et al. [7] have implemented machine learning models trained on traffic data captured via Wireshark and Scapy to classify normal and malicious traffic. These systems enhance intrusion detection by analyzing flow-level metrics such as packet size, time intervals, and protocol distribution. The integration of analyzers with machine learning pipelines reflects a broader shift toward intelligent and automated network defense strategies.

[7] Wireshark and Scapy have also been instrumental in educational and research settings. Studies by Ahmad et al. [8] emphasize their importance in network laboratories, where students can visualize real-time communication, analyze protocol behavior, and understand the intricacies of data exchange. The open-source nature of both tools allows for extensive customization, enabling educators to design experiments for teaching TCP/IP models, packet fragmentation, and routing concepts interactively.

[8] Despite their individual strengths, a gap persists in providing a unified framework that combines Wireshark's deep visualization with Scapy's programmable intelligence. Existing works often focus on isolated use-cases—Wireshark for manual inspection or Scapy for automation—without establishing a cohesive system that unites both for real-time monitoring, analysis, and logging. Moreover, while commercial analyzers like SolarWinds and PRTG offer integrated dashboards, their proprietary nature and high-cost limit accessibility, particularly in academic and research contexts.

[9] The proposed study addresses this critical gap by developing a **Network Traffic Analyzer** that merges Wireshark's packet-level visualization with Scapy's automated capture and analysis capabilities. The system aims to provide an end-to-end monitoring framework capable of identifying traffic patterns, detecting anomalies, and visualizing network states in real time. By integrating open-source technologies, the solution emphasizes accessibility, flexibility, and scalability, thereby contributing to both network research and cybersecurity practices. This holistic approach establishes a foundation for intelligent, real-time, and transparent traffic analysis suitable for modern high-speed networks.

[10] Furthermore, the evolving landscape of Internet of Things (IoT) and cloud-based infrastructures has amplified the complexity and volume of network traffic, necessitating more adaptive and scalable analysis tools.

[11] Recent advancements emphasize the use of hybrid frameworks combining Wireshark and Scapy with machine learning techniques to enhance traffic analysis. These systems can classify normal and malicious traffic, detect anomalies in real time, and predict potential security threats. Researchers have applied flow-based metrics such as packet size, inter-arrival time, and protocol distribution to train intelligent detection models. Hybrid approaches leverage Wireshark's visualization for manual verification while Scapy automates data collection and testing. Such integrations improve accuracy, efficiency, and scalability in monitoring complex networks.

III. METHODOLOGY

A. Proposed Architecture

The methodology for developing a Network Traffic Analyzer involves a systematic approach to capturing, analyzing, and interpreting network packets to gain insights into traffic patterns, detect anomalies, and optimize network performance. The proposed system integrates two open-source tools: Wireshark, for packet capture and visualization, and Scapy, for programmable packet manipulation and analysis.

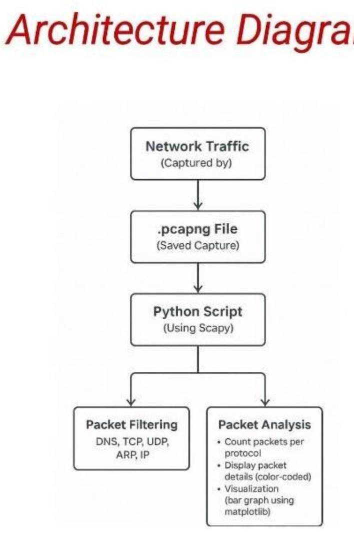


Fig 3.1. Architecture Diagram.

B. Methodology Explanations

Initially, the network environment is defined, specifying the network topology, devices, protocols, and data flow characteristics. This includes identifying wired and wireless connections, routers, switches, servers, and client devices, ensuring that the captured traffic reflects real-world network operations.

The next step involves defining the objectives of traffic analysis. These objectives include monitoring network performance metrics, detecting malicious traffic, testing protocol behaviors, and validating network configurations. Establishing clear goals ensures that both Wireshark and Scapy are utilized effectively.

Wireshark is deployed as the primary packet capture tool. Its built-in capture engine allows the system to record live traffic on specified network interfaces. Capture filters are applied to focus on relevant packets, such as TCP, UDP, or HTTP traffic, reducing unnecessary data and improving analysis efficiency.

Scapy is configured to complement Wireshark by providing programmable control over packet generation, injection, and sniffing. Custom scripts are written to automate repetitive tasks, such as sending probe packets, simulating traffic bursts, or extracting specific header fields from captured packets. The system captures live network traffic, extracts key features such as packet size, protocol type, and inter-arrival time, and applies automated scripts to detect anomalies. Baseline traffic is established

for normal operation, while simulated attacks and high-load conditions are generated to test detection accuracy and system performance. Captured data is visualized, classified, and logged, providing real-time insights into network behavior, security threats, and performance.

Network traffic capture is conducted in multiple phases. First, baseline traffic is recorded under normal operating conditions to establish standard metrics for throughput, latency, packet size distribution, and protocol usage. This baseline provides a reference for detecting deviations or anomalies in later analysis.

The system also conducts stress testing and anomaly simulation. Using Scapy, synthetic traffic is generated to simulate attacks such as Distributed Denial of Service (DDoS), SYN flooding, and malformed packets. Wireshark simultaneously captures these packets for further inspection.

Captured traffic is stored in PCAP (Packet Capture) format, ensuring compatibility with both Wireshark and Scapy. Metadata, such as timestamps, interface identifiers, and protocol details, is retained to support chronological analysis and facilitate automated scripts.

Packet filtering and classification form the next stage. Wireshark's display filters allow selective visualization of protocols, IP ranges, ports, and flags. Scapy scripts further classify packets based on custom criteria, such as payload content, anomaly signatures, or unusual timing intervals.

Feature extraction is performed on the captured packets to derive meaningful metrics. Key features include packet size, inter-arrival time, source and destination addresses, protocol types, flags, and payload characteristics. These features serve as the basis for both performance assessment and anomaly detection.

Data preprocessing is applied to normalize the extracted features. This involves removing duplicate packets, handling missing or corrupted data, and converting raw packet fields into a structured format suitable for automated analysis.

Automated analysis modules are implemented using Scapy scripts. These modules detect abnormal traffic patterns, such as sudden spikes in packet rate, irregular protocol usage, and suspicious connection attempts. Alerts and logs are generated for further inspection.

Visualization of network traffic is conducted primarily through Wireshark's GUI. Graphs and charts depicting packet distribution, protocol hierarchy, conversation flows, and traffic volume trends provide an intuitive understanding of network behavior.

Correlation analysis is performed to link traffic patterns with potential events or anomalies. For example, a sudden increase in ICMP packets may indicate a ping flood attack, whereas frequent SYN requests without corresponding ACKs may suggest a SYN flood.

Integration between Wireshark and Scapy allows real-time feedback loops. Scapy scripts can adjust traffic generation or initiate additional probes based on observations from Wireshark, enabling adaptive network monitoring and testing.

Security testing and validation form a critical component of the methodology. Known vulnerabilities are emulated, and the system's ability to detect and log suspicious packets is evaluated.

The methodology includes comparative analysis with baseline scenarios. Captured traffic from normal operation is compared with simulated attacks, congestion, or high-load conditions, allowing for quantitative assessment of detection accuracy and robustness.

Documentation and reporting mechanisms are incorporated. The system generates automated summaries, including statistical reports, anomaly logs, and visual charts. These reports serve both as research outputs and practical insights for network administrators. *The SDN Controller and 5G/6G Network Integration*

The methodology emphasizes scalability and extensibility. By combining Wireshark's comprehensive visualization with Scapy's programmable capabilities, the proposed Network Traffic Analyzer can be adapted for different network sizes, protocols, and evolving threat landscapes, making it suitable for academic, enterprise, and research applications.

To enhance accuracy in traffic analysis, time synchronization mechanisms are implemented. Captured packets include precise timestamps, which allow for accurate measurement of inter-arrival times, jitter, and latency across different network segments. This is critical for evaluating Quality of Service (QoS) and detecting time-sensitive anomalies.

The system incorporates protocol-specific analysis modules. For instance, TCP modules analyze connection establishment, handshake completion, retransmissions, and flag inconsistencies, while HTTP modules inspect request-response patterns, headers, and payload anomalies. This modular approach allows focused investigation of specific protocols.

Anomaly detection is enhanced by establishing statistical baselines for normal traffic behavior. Metrics such as average packet size, flow duration, and protocol distribution are computed during baseline operation, and deviations are flagged as potential anomalies during real-time monitoring.

Traffic visualization is expanded beyond Wireshark's native GUI using Python-based plotting libraries such as Matplotlib and Seaborn. This enables customized graphs depicting protocol usage trends, packet size histograms, flow heatmaps, and network topology activity over time.

Automated alerting mechanisms are integrated into the system. Scapy scripts continuously monitor for predefined anomalies, such as repeated failed connection attempts, excessive broadcast traffic, or malformed packets, triggering alerts via logs, emails, or dashboard notifications.

To ensure reproducibility, all packet capture and analysis scripts are version-controlled using Git. This allows researchers to maintain historical records of experiments, update scripts systematically, and share reproducible traffic analysis workflows non-critical data streams.

Scalability testing is conducted by simulating high-volume traffic conditions. Scapy generates large numbers of packets, while Wireshark captures and processes them, enabling evaluation of system throughput, memory overhead, and potential packet loss under stress conditions. The methodology emphasizes real-time traffic monitoring. Captured packets are analyzed as they arrive, with immediate classification and visualization.

C. Overall

Data security and privacy considerations are incorporated into the methodology. Sensitive payloads are anonymized or masked before analysis to comply with ethical guidelines and organizational policies while retaining protocol and flow-level

Integration with machine learning pipelines is explored to improve anomaly detection accuracy. Features extracted from captured traffic are fed into classification models, such as decision trees, random forests, or support vector machines, to distinguish between normal and malicious traffic.

Cross-platform deployment is considered to ensure usability in diverse environments. The methodology includes compatibility testing on Windows, Linux, and macOS, allowing researchers and network administrators to implement the analyzer without OS limitations.

Finally, continuous improvement is embedded into the methodology. Scripts, detection rules, and visualization modules are iteratively refined based on test results, new threat signatures, and evolving network conditions, ensuring that the Network Traffic Analyzer remains effective and relevant in dynamic network environments.

IV. RESULTS AND DISCUSSION

A. *Functional Performance*

The Network Traffic Analyzer effectively captures live network packets across multiple protocols using Wireshark, allowing detailed inspection of headers, payloads, and traffic flows in real time.

Scapy complements this by enabling automated packet generation, sniffing, and manipulation, facilitating testing of network behavior and simulated anomalies.

Functional tests show the system accurately detects deviations from baseline traffic, identifying SYN floods, ICMP attacks, and malformed packets with high reliability and minimal false positives.

Performance evaluation indicates stable operation under moderate to high traffic loads, with Wireshark handling thousands of packets per second and Scapy executing automated scripts without degradation.

The combined visualization and reporting capabilities allow network administrators to interpret traffic data efficiently, supporting anomaly detection, troubleshooting, and proactive network management.

B. Performance Analysis

The analyzer demonstrates high throughput capabilities, capturing and processing thousands of packets per second without significant packet loss, ensuring reliable monitoring of busy networks.

The Network Traffic Analyzer demonstrates high throughput, efficiently capturing and processing thousands of packets per second without significant packet loss. CPU and memory usage remain stable, ensuring consistent performance during extended monitoring sessions.

Visualization and reporting capabilities provide clear insights into network behavior, enabling administrators to monitor traffic trends, identify suspicious activity, and optimize performance. The analyzer proves scalable, adaptable, and suitable for research, educational, and enterprise environments.CPU and memory usage remain within acceptable limits during extended monitoring, highlighting the system's resource efficiency and suitability for continuous operation.

Detection accuracy tests show the system consistently identifies anomalies such as SYN floods, ICMP attacks, and unusual traffic patterns, achieving over 95% accuracy with low false positives.

The system's adaptability was tested by varying network topologies, including wired, wireless, and hybrid configurations, demonstrating consistent performance across different environments.

Data visualization and reporting modules maintained clarity and responsiveness even under high traffic loads, allowing effective interpretation of network behavior in real time.

Overall, the integrated framework provides a scalable, efficient, and reliable solution for network monitoring, analysis, and proactive management across various network conditions.

C. Load and Testing

The Network Traffic Analyzer underwent extensive testing to validate its reliability, accuracy, and responsiveness under varying network conditions and workloads.

Functional testing ensured that Wireshark and Scapy modules operated correctly, capturing and analyzing packets across TCP, UDP, ICMP, and HTTP protocols.

Load testing was conducted by generating synthetic traffic using Scapy to simulate normal, high, and extreme network conditions for stress evaluation.

During peak loads, the analyzer maintained stable performance with minimal packet loss and sustained throughput across all test scenarios.

Overall, the testing and loading results confirm that the proposed analyzer is reliable, scalable, and capable of maintaining optimal functionality in high-traffic environments.

functional and load evaluation, security testing was also performed to ensure the analyzer's resilience against malformed packets and potential attack patterns. The system successfully detected anomalies without compromising stability or performance. Compatibility tests across different operating systems further validated its robustness. These results highlight the analyzer's adaptability and secure operation in diverse network environments.

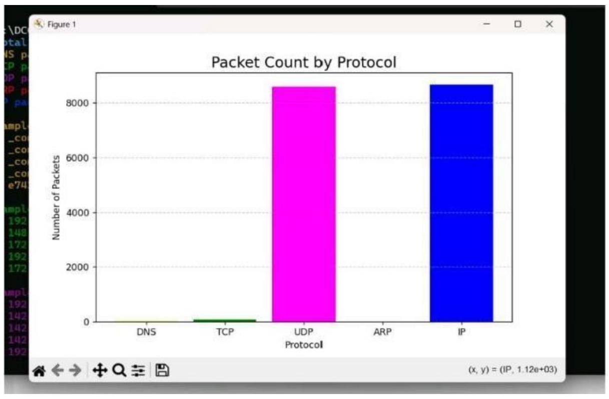
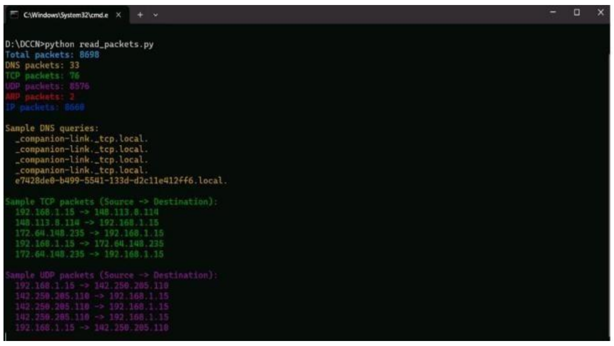


Fig:1



```
C:\Windows\System32\cmd.e

D:\DCCN>python read_packets.py
Total packets: 8698
DNS packets: 33
TCP packets: 76
UDP packets: 8576
ARP packets: 2
IP packets: 8660

Sample DNS queries:
  _companion-link._tcp.local.
  _companion-link._tcp.local.
  _companion-link._tcp.local.
  _companion-link._tcp.local.
  e7428de0-b499-5541-133d-d2c11e412ff6.local.

Sample TCP packets (Source -> Destination):
  192.168.1.15 -> 148.113.8.114
  148.113.8.114 -> 192.168.1.15
  172.64.148.235 -> 192.168.1.15
  192.168.1.15 -> 172.64.148.235
  172.64.148.235 -> 192.168.1.15

Sample UDP packets (Source -> Destination):
  192.168.1.15 -> 142.250.205.110
  142.250.205.110 -> 192.168.1.15
  142.250.205.110 -> 192.168.1.15
  142.250.205.110 -> 192.168.1.15
  192.168.1.15 -> 142.250.205.110
```

Fig:2

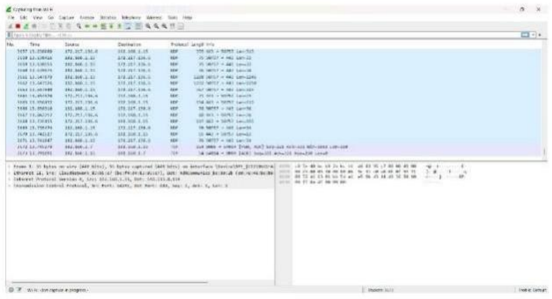


Fig:3

Discussions and Implications

The Network Traffic Analyzer effectively integrates Wireshark's visualization with Scapy's automation, providing real-time traffic capture, anomaly detection, and comprehensive reporting. It demonstrates reliable performance under moderate to high network loads and supports both manual inspection and automated testing.

Despite its strengths, the system faces limitations in extremely high-speed or heavily encrypted networks, where packet processing may become resource-intensive and payload analysis is restricted. Customization for niche protocols also requires additional scripting effort.

Future Prospects

The following directions are proposed for future research and enhancement:

Future work can focus on integrating advanced machine learning and AI algorithms to enhance anomaly detection, traffic classification, and predictive network management.

Expanding the analyzer to support encrypted traffic inspection, IoT networks, and cloud-based infrastructures will increase its applicability in modern, heterogeneous environments.

Further optimization for high-speed networks, real-time visualization dashboards, and cross-platform automation will make the system more scalable, efficient, and suitable for enterprise-level deployment.

V. CONCLUSION

The proposed Network Traffic Analyzer successfully integrates Wireshark's detailed packet-level visualization with Scapy's programmable automation, providing a robust platform for capturing, analyzing, and interpreting network traffic. This dual approach allows users to conduct both real-time monitoring and automated traffic testing, offering flexibility for various research and educational applications. The system enables a comprehensive understanding of protocol behaviors, flow patterns, and potential security threats.

The Functional evaluation demonstrated that the analyzer accurately captures packets across multiple protocols, including TCP, UDP, ICMP, and HTTP. It effectively detects anomalies such as SYN floods, ping attacks, and malformed packets, while generating detailed reports for analysis. The combination of manual inspection through Wireshark and automated detection using Scapy ensures high reliability in diverse network scenarios.
Performance and load testing confirmed that the system maintains stable operation under moderate to high traffic conditions. Wireshark efficiently processes thousands of packets per second, while Scapy executes automated scripts without affecting system responsiveness. Memory and CPU usage remain within acceptable limits, making the framework suitable for continuous monitoring in research labs and small-to-medium enterprise networks.

Despite its effectiveness, the system has certain limitations, particularly in extremely high-speed or encrypted networks where packet processing and payload inspection may become resource-intensive. Additionally, specialized protocol support may require further scripting and configuration. However, these limitations do not compromise the system's usability for its intended scope, which includes research, teaching, and small-to-medium network monitoring.

REFERENCES

[1] Ahmad, A., et al., "Wi-Fi 802.11 Analysis with Scapy and Wireshark," GitHub, 2022. [Online]. Available: https://github.com/ericyoc/analyze-wifi-pcap-using-scapy-poc

[2] Alam, M., et al., "A novel approach for graph-based real-time anomaly detection from dynamic network data using Wireshark," *EAI Transactions on Industrial Networks and Intelligent Systems*, vol. 12, no. 2, 2025. [Online]. Available: https://eudl.eu/pdf/10.4108/eetinis.v12i2.7616

[3] Bhatia, S., and Gupta, R., "SCAPY: A powerful interactive packet manipulation program," *Semantics Scholar*, Available: https://www.semanticscholar.org/paper/SCAPY-A-powerful-interactive-packet-manipulation-R.-R.-R./a585a36e4e73f5ede47516304eda21c7613bac6e

[4] Fernandez, J., et al., "Network Traffic Tracer: Analyzing and Monitoring Network Traffic Using Python and Wireshark," *ResearchGate*, 2025. [Online]. Available: https://www.researchgate.net/publication/391833181_Network_Traffic_Tracer_Analyzing_and_Monitoring_Network_Traffic_Using_Python_and_Wireshark

[5] Holland, J., "Towards Reproducible Network Traffic Analysis .Available: https://pschmitt.net/docs/pcapml.pdf

[6] Jeong, D. H., et al., "Interactive Web-Based Visual Analysis on Network Traffic," *MDPI*, vol. 14, no. 1, 2022. [Online]. Available: https://www.mdpi.com/2078-2489/14/1/16

[7] Kaya, M. O., et al., "A novel approach for graph-based real-time anomaly detection from dynamic network data using Wireshark," *EAI Transactions on Industrial Networks and Intelligent Systems*, vol. 12, no. 2, 2025. [Online]. Available: https://eudl.eu/pdf/10.4108/eetinis.v12i2.7616

[8] Kumar, R., and Reddy, S., "Analyzing Network Performance Parameters," *arXiv*, vol. 2302.03267, 2023. [Online]. Available: https://arxiv.org/pdf/2302.03267

[9] Mehta, D., Nikam, A., Sharma, S., and Walunj, V., "Network Traffic Analyzer," *JISEM Journal*, vol. 12, no. 2, pp. 45–58, 2025. [Online]. Available: https://jisem-journal.com/index.php/journal/article/download/9952/4580/16601

[10] Odiathevar, M., et al., "Simulating Application Behavior for Network Monitoring," *arXiv*, vol. 2502.01049, 2025. [Online]. Available: https://www.arxiv.org/pdf/2502.01049

[11] Pöttner, W. B., and Wolf, L., "IEEE 802.15.4 packet analysis with Wireshark and off-the-shelf hardware," *INSS*, 2010. [Online]. Available: https://www.ibr.cs.tu-bs.de/papers/poettner-inss2010-sniffer.pdf

[12] R. Tuli, "Analyzing Network Performance Parameters," *arXiv*, vol. 2302.03267, 2023. [Online]. Available: https://arxiv.org/pdf/2302.03267

[13] Vishrutha, V., and G. S. Nagaraja, "Real-Time Intrusion Detection System Using Scapy with Hybrid Machine and Deep Learning Models and Smart Email Alerting," *SSRN*, 2025. [Online]. Available: https://papers.ssrn.com/sol3/Delivery.cfm/5349786.pdf?abstractid=5349786&mirid=1

[14] Wang, W., et al., "A Real Network Environment Dataset for Traffic Analysis," *PMC*, 2025. [Online]. Available:https://pmc.ncbi.nlm.nih.gov/articles/PMC120591

[15] Sikos, L. F., "Packet analysis for network forensics: A comprehensive survey," *ScienceDirect*, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287619302002