Network Traffic Analysis Using Machine Learning

Gari Ajay
Reg. No. 24Q71F0015
gariajaykumar@gmail.com
Department of Master of Computer Applications
Avanthi Institute of Engineering and Technology (Autonomous)
Vizianagaram, Andhra Pradesh, India
Under the guidance of Mr. punnana Chandrasekhar rao, MCA, Assistant Professor
punnanachandrasekhar@gmail.com

Abstract*—In the world of networking, it sometimes becomes essential to know what types of applications flow through the network for the performance of certain tasks. Network-traffic classification is widely used among Internet Service Providers to analyse the characteristics required to design the network, which directly affects the overall performance of a network. Various techniques are adopted to classify network protocols, including port-based, payload-based, and machine-learning-based approaches, each with their own advantages and limitations. Machine learning has become prominent due to its growing use in other fields and its reported accuracy compared with traditional methods. In this work, two basic algorithms, Naïve Bayes and K-Nearest Neighbours (KNN), are compared along with Decision Tree and Support Vector Machine, when applied to a networking dataset extracted from live video feeds using Wireshark. The machine-learning implementation uses the Python scikit-learn library together with NumPy and pandas as helper libraries, and Wireshark is used as the packet-capture and protocol-analysis tool. The reported observation from the source work is that K-Nearest Neighbours provided more accurate predictions than Naïve Bayes, Decision Tree, and Support Vector Machine on the captured dataset. The system was validated through a comprehensive functional test plan covering normal-traffic classification, malicious-traffic detection, unknown-attack detection, feature extraction, encrypted-traffic handling, imbalanced-data handling, real-time analysis, noise robustness, model generalisation, attack-type classification, data drift, false-positive minimisation, throughput stress, retraining, adversarial resilience, protocol diversity, session-based analysis, and label-quality checks. The work provides a practical reference for ISPs and network administrators evaluating ML-based traffic classification.*

Keywords*—Network Traffic Analysis; Machine Learning; K-Nearest Neighbours; Naïve Bayes; Decision Tree; Support Vector Machine; Wireshark; Packet Sniffing.*

I. INTRODUCTION

Modern networks carry an enormous variety of applications, and knowing what types of traffic flow through the network is essential for performance tuning, capacity planning, troubleshooting, and security. Network-traffic classification is widely used among Internet Service Providers (ISPs) and network administrators to analyse traffic characteristics and design and operate networks effectively. With the

introduction of web browsers and HTML in the early 1990s, and the rapid expansion of internet-based services in the decades that followed, modern networking now spans wired, wireless, mobile, and cloud environments, and properly configured networks rely on firewalls, encryption, and access controls to protect data from unauthorized access.

Network traffic can be classified using several techniques. Port-based classification relies on well-known port numbers but is unreliable when applications use non-standard ports, payload-based classification inspects packet contents but is sensitive to encryption and privacy concerns, and machine-learning-based classification learns patterns from traffic features and has become prominent because of its accuracy and adaptability. Packet sniffing—capturing, analysing, and interpreting network packets as they traverse a network—is a fundamental tool for monitoring, troubleshooting, and traffic logging, and Wireshark, an open-source packet analyser, allows network administrators to capture live data at a microscopic level for analysis.

In this project, network traffic is captured using Wireshark, processed in Python, and classified using machine-learning algorithms. Naïve Bayes, K-Nearest Neighbours, Decision Tree, and Support Vector Machine algorithms are compared on the captured dataset. The Python implementation uses the scikit-learn library along with NumPy and pandas, and Wireshark provides the underlying packet capture and protocol decoding. The objective is to compare these basic algorithms and identify which performs best on the captured traffic, and to provide actionable insights for optimising network performance and security.

II. LITERATURE SURVEY

Several studies have examined the use of Wireshark and machine learning for network-traffic analysis. Bindu Dodiya et al. demonstrated the use of Wireshark as an efficient open-source packet-analysis tool for tracing and categorising attack signatures in network forensics, offering detailed packet-level visibility for proactive cybersecurity, while noting that Wireshark itself does not provide intrusion-detection capabilities and must be complemented by additional measures. Giovanni Barbieri et al. contrasted Shodan-only assessments with large-scale traffic analysis at an Internet Exchange Point using sFlow sampling, enabling the identification of Industrial Control System endpoints engaged in genuine industrial traffic and differentiating industrial from IT traffic, with the limitation that the 31-day sampled capture might miss transient patterns. Muhammad Farrid Affiq Harirul Kamal et al. proposed a dynamic Android botnet-detection method that extracts five features from Wireshark and SSL packet capture and reports promising accuracy, ROC and false-positive performance with an Artificial Neural Network.

Sujith Bebortta et al. focused on network-traffic administration for diverse IoT devices through packet-level and flow-level inter-arrival-rate analysis, supporting identification and management of IoT devices. Ali Siddiqui et al. used Wireshark as a network-protocol analyser for forensic analysis of network-security attacks, supporting detailed analysis of HTTP, TCP, and UDP, while Mohammed Al Fawar et al. emphasised the role of traffic analysis for optimising performance and securing networks. Laura Chappell et al. proposed distributed network analysis using exported packet records imported to the real-time tool TOPAS and analysed via Wireshark, and Banerjee, Usha et al. described Wireshark as a sniffing tool whose effectiveness for malicious-packet detection suggests potential for development into a robust intrusion-detection system. Samer Hamdani et al. reported a CoAP-vs-MQTT comparison study relevant to IoT data

transmission, Gerhard Munz et al. proposed a low-cost distributed network-analysis approach using PSAMP/NetFlow exports to TOPAS and Wireshark, and Pallavi Asrodia et al. emphasised the importance of packet sniffers for monitoring wired and wireless networks. This body of work motivates an ML-based traffic-classification system using Wireshark for capture and Python's scikit-learn ecosystem for modelling.

TABLE I. SUMMARY OF REPRESENTATIVE PRIOR WORK

S.No	Author(s)	Contribution	Note / Limitation
1	Dodiya et al.	Wireshark for attack-signature forensics	No built-in intrusion detection
2	Barbieri et al.	Shodan vs. IXP sFlow analysis	Limited sampling window
3	Kamal et al.	Android botnet detection (ANN)	Dataset diversity limits
4	Bebortta et al.	Packet/flow analysis for IoT	No proactive isolation
5	Siddiqui et al.	Wireshark for forensic analysis	Coarse secure/vulnerable labels
6	Chappell et al.	Distributed analysis via TOPAS+Wireshark	Cost-aware alternative
7	Munz et al.	PSAMP/NetFlow + TOPAS	Scalable, infrastructure-based
8	Asrodia et al.	Packet sniffers in monitoring	Foundational survey

III. EXISTING SYSTEM AND PROPOSED SYSTEM

A. Existing System

Existing network-traffic-analysis approaches commonly rely on port-based and payload-based classification or on raw packet capture and manual inspection using tools such as Wireshark. Port-based classification is fast but unreliable for applications that use non-standard or randomised ports, and payload-based classification is sensitive to encryption and raises privacy concerns. Manual Wireshark analysis is powerful for forensic and troubleshooting work but is labour-intensive at scale and does not, by itself, provide intrusion detection or automated classification. Existing ML-based studies often focus on a single algorithm or a single domain, leaving open the basic comparison of common classifiers on a Wireshark-captured dataset.

Limitations of the existing system:

- Port-based classification fails on non-standard ports.
- Payload-based classification is limited by encryption and privacy.
- Manual Wireshark analysis is labour-intensive at scale.
- Wireshark alone offers no intrusion detection or auto-classification.
- Many existing ML studies focus on single algorithms or domains.

B. Proposed System

The proposed system captures network traffic using Wireshark from a live video feed, exports the data for analysis, and classifies it using machine-learning algorithms implemented in Python with scikit-learn. Four basic algorithms—Naïve Bayes, K-Nearest Neighbours, Decision Tree, and Support Vector Machine—are compared on the same dataset to identify the best performer for the captured traffic. Wireshark provides advanced filtering, deep packet inspection, statistical profiling, and protocol analysis, while the ML pipeline performs preprocessing, feature extraction, training, evaluation, and prediction.

Advantages of the proposed system:

- Combines Wireshark capture with ML-based classification.
- Direct comparison of KNN, Naïve Bayes, Decision Tree, and SVM.
- Uses standard, accessible tools (Python, scikit-learn, NumPy, pandas).
- Supports filtering, decoding, flow analysis, and statistical profiling.
- Provides actionable insights for performance and security.
- Extensible to additional algorithms and traffic sources.

IV. SYSTEM DESIGN AND METHODOLOGY

A. Data Capture and Preparation

Network traffic is captured using Wireshark, which provides comprehensive packet-level visibility and protocol decoding. Captured packets are filtered to eliminate noise and isolate packets of interest, using filters based on IP addresses, port numbers, or specific protocols. Packets are grouped into flows or connections by source–destination pairs and protocol, which supports the tracking of communication patterns and the identification of abnormal behaviour. The filtered data is exported and ingested into the ML pipeline using pandas.

B. Feature Engineering and Models

From the captured traffic, relevant features such as IP addresses, port numbers, protocol type, packet size, and flow-duration statistics are extracted to form the training data. The dataset is split into training and testing subsets, and four classifiers are trained on the same features for direct comparison: Naïve Bayes, K-Nearest Neighbours, Decision Tree, and Support Vector Machine. The models are evaluated on the held-out data using metrics such as accuracy, precision, recall, F1-score, and confusion matrix, and the best-performing model is selected for downstream use.

C. Workflow

Wireshark captures traffic; the captured PCAP/CSV files are filtered and exported; the ML pipeline preprocesses the data, extracts features, trains the four classifiers, and evaluates them; results are presented as classification reports and visualisations such as confusion matrices and feature-importance summaries; and outcomes guide actionable insights for the network operator.

V. SYSTEM IMPLEMENTATION

A. Technology Stack

TABLE II. TECHNOLOGY STACK

Component	Technology / Tool
Programming Language	Python
Packet Capture / Analysis	Wireshark
ML Library	scikit-learn
Numerical / Data	NumPy, pandas
Algorithms	Naïve Bayes, K-Nearest Neighbours, Decision Tree, SVM
Data Source	Live network traffic captured via Wireshark
Outputs	Predicted class, metrics, confusion matrix, charts

B. Implementation Details

The system is implemented in Python. The ML implementation uses the scikit-learn library, with NumPy and pandas as helper libraries for numerical computation and tabular data handling. Wireshark captures traffic at the packet level and supports advanced filtering, deep packet inspection, statistical profiling, and protocol analysis; representative analysis steps include data filtering, packet decoding (e.g., HTTP requests or DNS queries), flow analysis grouping by source–destination and protocol, statistical profiling of packet counts and size distributions, and protocol-specific analysis such as TCP handshake inspection. The classifiers are trained on the extracted features and compared on the same train/test split.

C. Algorithm Comparison

Naïve Bayes is fast and works well when features are reasonably independent, but can be inaccurate when this assumption is violated. K-Nearest Neighbours is simple and interpretable but its computational cost grows with dataset size because the distance from a query point to every existing point must be computed, which can degrade performance on large datasets. Decision Tree provides interpretable rule-based classification but can overfit, and Support Vector Machine handles complex decision boundaries but can be sensitive to parameter tuning and kernel choice. On the dataset captured for this work, the source observes that KNN produced more accurate predictions than Naïve Bayes, Decision Tree, and SVM; this qualitative result is reported here without claiming specific numeric accuracy values that are not provided in the source.

VI. SYSTEM TESTING AND RESULTS

Testing was carried out through unit testing, integration testing, and a comprehensive functional plan covering nineteen test cases. The plan addresses normal-traffic classification, malicious-traffic detection (DDoS), unknown-attack detection, feature-extraction validation, encrypted-traffic handling, imbalanced-data handling, real-time analysis, noise robustness, feature-importance validation, model generalisation

across datasets, multi-class attack classification (DoS / Probe / R2L / U2R), data-drift detection, false-positive minimisation, throughput stress testing, model retraining, adversarial resilience, protocol diversity, session-based sequence analysis, and label-quality checks. The reported test results state that all defined test cases passed successfully and no defects were encountered.

TABLE III. REPRESENTATIVE TEST CASES

ID	Scenario	ML Task	Expected Outcome	Metric
TC-01	Normal-traffic classification	Classification	Labels benign traffic correctly	Accuracy, Precision
TC-02	Malicious-traffic detection	Classification	Identifies DDoS traffic	Recall, F1
TC-03	Unknown-attack detection	Anomaly detection	Flags unknown patterns	ROC-AUC, FPR
TC-05	Encrypted-traffic handling	Classification	Uses metadata, not payload	Accuracy
TC-07	Real-time traffic analysis	Online learning	Low-latency processing	Latency, Throughput
TC-11	Attack-type classification	Multi-class	Distinguishes DoS/Probe/R2L/U2R	Confusion matrix, F1
TC-16	Adversarial resilience	Robustness	Resists evasion attempts	Accuracy under attack

A. Observed Results

The implemented system captures network traffic using Wireshark, prepares features, trains four classical machine-learning models, and evaluates them on the same dataset. The source reports that KNN provided the most accurate predictions on the captured data, with Naïve Bayes, Decision Tree, and SVM serving as comparative baselines. Functional testing across the nineteen scenarios passed without defects, and the system supports filtering, decoding, flow analysis, statistical profiling, and protocol analysis through Wireshark. The source describes these outcomes qualitatively; no specific numeric accuracy values are claimed here, and real-world performance depends on dataset size, balance, and the diversity of traffic captured.

Representative screenshots from the prototype implementation:

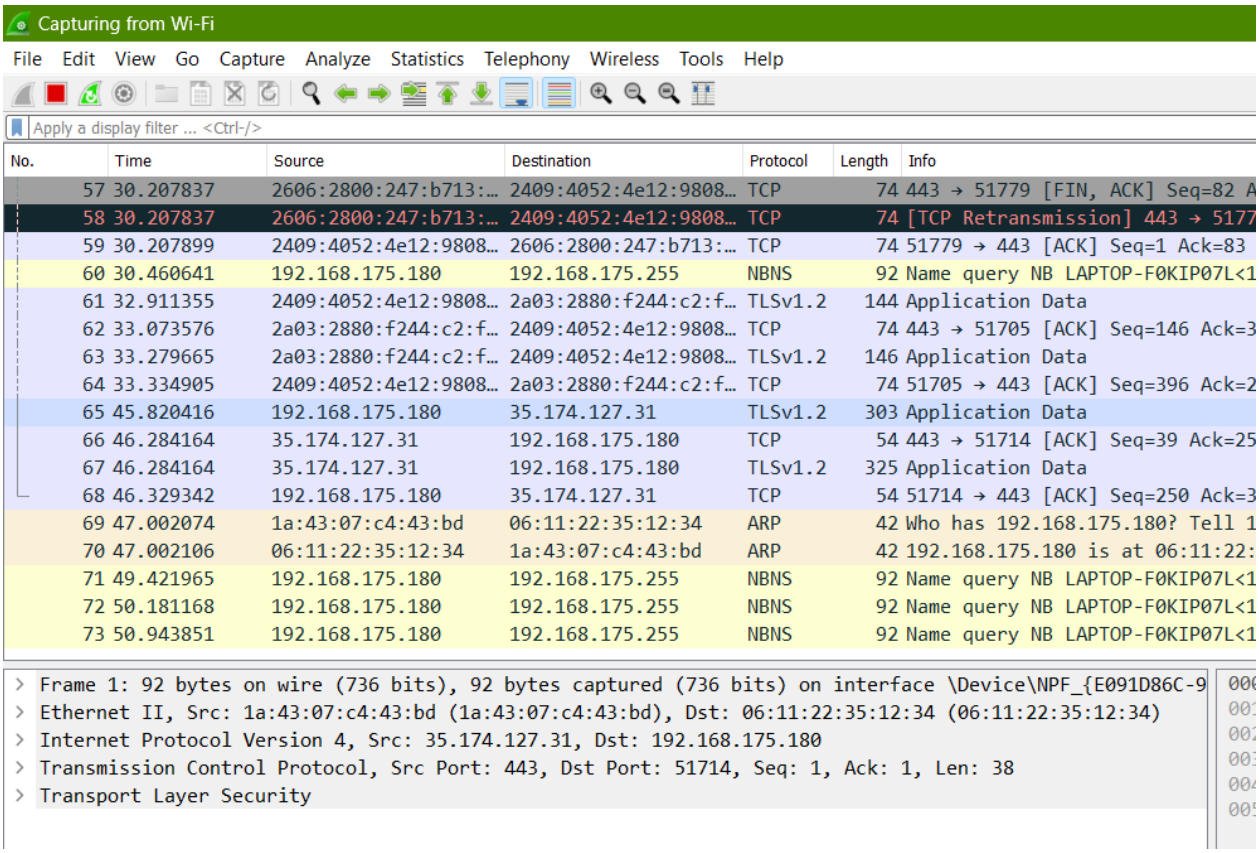

Fig. 1. Data filtering using Wireshark.

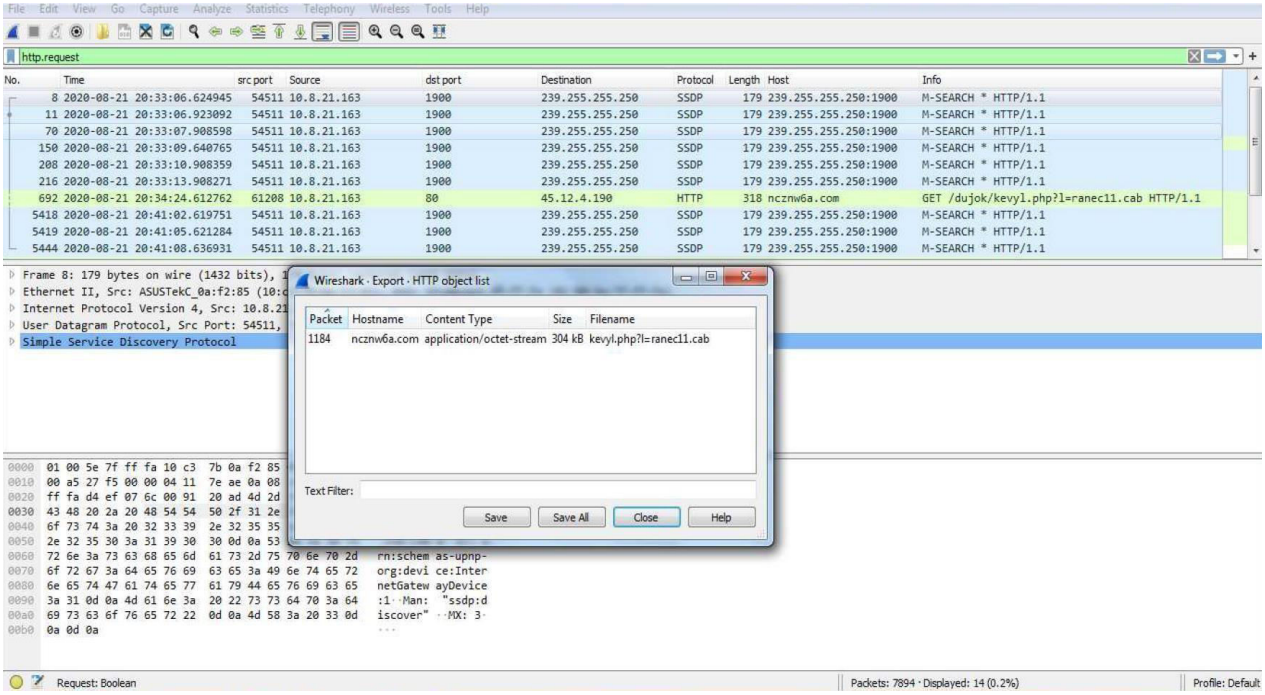

Fig. 2. Flow analysis using Wireshark.

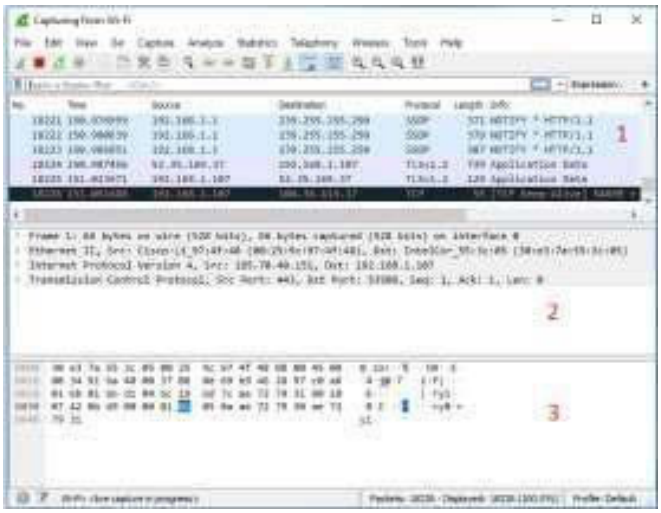

Fig. 3. Protocol analysis (HTTP / TCP / UDP).

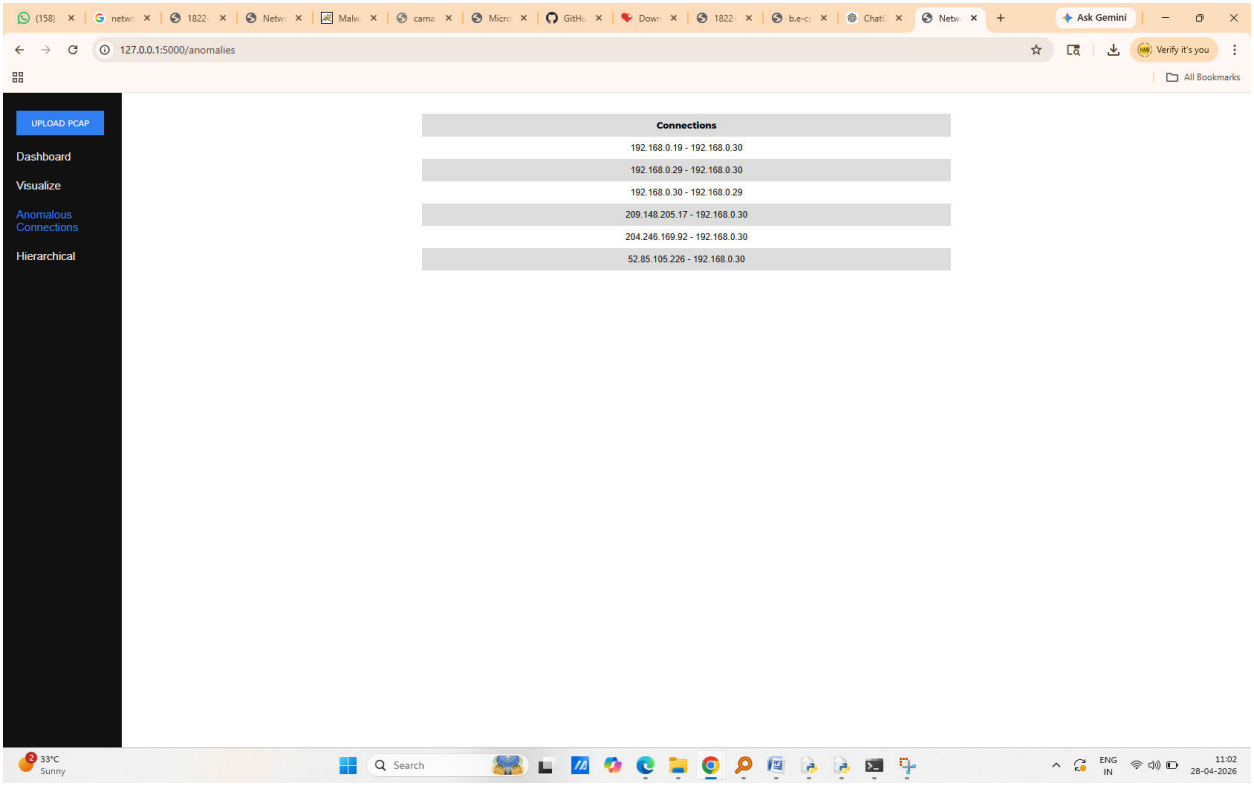

Fig. 4. Classifier comparison (KNN / Naïve Bayes / DT / SVM).

VII. CONCLUSION AND FUTURE SCOPE

Packet sniffing is useful for analysing data during transmission in the network, and tools such as Wireshark make it practical for monitoring, troubleshooting, traffic analysis, and forensic investigations. This project combined Wireshark-based packet capture with machine-learning classification implemented in Python (scikit-learn, NumPy, pandas) and compared Naïve Bayes, K-Nearest Neighbours, Decision Tree, and Support Vector Machine on a captured dataset; the source reports that KNN produced the most accurate predictions of the four algorithms on this data. Sniffing is straightforward in non-switched networks and harder on switched networks, where additional techniques are required, and Wireshark also supports rich filtering, decoding, flow and protocol analysis, and statistical profiling. The system thus provides a practical, end-to-end view of how ML can be applied to network-traffic classification and how the results can be used to improve performance and security.

Future enhancements include integrating real-time machine-learning anomaly and threat detection on top of Wireshark capture, extending the analysis to Internet of Things environments and cloud-based infrastructures, developing automated incident-response systems that combine Wireshark with other security tools, and integrating with big-data analytics platforms for large-scale traffic insight. Making the analysis pipeline platform-independent, adding neural-network-based classifiers, and providing performance-optimisation plugins are also valuable directions for production use.

REFERENCES

[1] A. Dabir et al., “Bottleneck Analysis of Traffic Monitoring Using Wireshark,” in Proc. Innovations in Information Technologies (IIT), Dubai, UAE, 2007, pp. 158–162.

[2] Banerjee, Usha et al., “Evaluation of Wireshark as a Sniffing Tool for Detection of Malicious Packets,” International Journal of Computer Applications.

[3] Pallavi Asrodia et al., “Network Traffic Analysis Using Packet Sniffer,” International Journal of Engineering Research and Applications, 2012.

[4] G. Munz et al., “A Low-Cost Method for Distributed Network Analysis,” Internet Performance Monitoring Workshop.

[5] Laura Chappell et al., Wireshark Network Analysis: The Official Wireshark Certified Network Analyst Study Guide.

[6] Samer Hamdani et al., “Comparative Study of CoAP vs. MQTT,” IoT Communications Research.

[7] Ankita Gupta, Kavita, and Kirandeep Kaur, “Vulnerability Assessment and Penetration Testing,” International Journal of Engineering Trends and Technology, vol. 4, no. 3, 2013.

[8] A. Vikram et al., “Blockchain Technology and Its Impact on Future of Internet of Things (IoT) and Cyber Security,” in Proc. Int. Conf. Electronics, Communication and Aerospace Technology (ICECA), 2022, pp. 444–447.

[9] V. Dharani et al., “Spam SMS (or) Email Detection and Classification Using Machine Learning,” in Proc. Int. Conf. Smart Systems and Inventive Technology (ICSSIT), 2023, pp. 1104–1108.

[10] Upendra Shetty D R and A. Patil et al., “Malicious URL Detection and Classification Analysis Using Machine Learning Models,” in Proc. Int. Conf. Intelligent Data Communication Technologies and IoT (IDCIoT), 2023, pp. 470–476.

[11] F. Khan, R. Kothari, M. Patel, and N. Banoth, “Enhancing Non-Fungible Tokens for the Evolution of Blockchain Technology,” in Proc. Int. Conf. Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2022, pp. 1148–1153, doi: 10.1109/ICSCDS53736.2022.9760849.

[12] D. Kothari, M. Patel, and A. K. Sharma, “Implementation of Grey Scale Normalization in Machine Learning & AI for Bioinformatics Using CNNs,” in Proc. 6th Int. Conf. Inventive Computation Technologies (ICICT), Coimbatore, India, 2021, pp. 1071–1074, doi: 10.1109/ICICT50816.2021.9358549.