Intechno Journal :
Information Technology Journal
Vol. 8 No. 1 July 2026
Homepage: https://jurnal.amikom.ac.id/index.php/intechno
INTECHNO Journal
INFORMATION TECHNOLOGY JOURNAL
e-ISSN: 2655-1438
p-ISSN: 2655-1632

.

An Empirical Benchmarking Framework for IoT Traffic Anomaly Detection Using Elastic Stack SIEM

Ferdiansyah[1], Reynaldi Rizki Billanivo[2], M Ardiansyah[3], Tegar Putra[4], M. Habibullah Amin[5]

[1,2,3,4,5]Faculty of Computer and Science, Universitas Indo Global Mandiri Palembang, Indonesia

Article Info

Article history:

Received: 12 June 2026
Revised: 29 June 2026
Accepted: 2 July 2026
Published: 31 July 2026

Keywords:

Machine Learning
Network Anomaly Detection
K-Nearest Neighbors
Support Vector Machine
Naive Bayes

ABSTRACT

Purpose: This study aims to construct a realistic IoT-MQTT benchmark dataset and evaluate supervised machine learning classifiers for detecting network traffic anomalies, specifically Distributed Denial of Service (DDoS) and spoofing attacks, within a live Security Information and Event Management (SIEM) environment.
Methods: An empirical benchmarking framework based on a live Elastic (ELK) Stack SIEM environment was developed, and supervised machine learning classifiers were evaluated for IoT network traffic anomaly detection.
Result: KNN and SVM achieved the highest accuracy (0.99), whereas Naive Bayes achieved 0.96. Further analysis revealed that the superior performance of KNN and SVM was largely influenced by data leakage caused by the _attacker_ip feature, while Naive Bayes demonstrated better generalization without relying on identity-based features.
Conclusion: The findings highlight the importance of rigorous feature engineering and data leakage analysis when developing machine learning models for IoT traffic anomaly detection, particularly in live SIEM environments. Moreover, the proposed framework contributes to the achievement of Sustainable Development Goals (SDGs) 9 by supporting resilient digital infrastructure and secure IoT-based innovation.

Corresponding Author:

Ferdiansyah
Email: ferdiansyah.it@gmail.com

1. INTRODUCTION

The rapid advancement of information and communication technologies has significantly expanded the adoption of computer networks across key sectors such as education, governance, business, and industry [1]. Computer networks function as the primary infrastructure for swift and efficient data transmission [2]. However, this interconnectivity also increases exposure to various cyber threats, including malware distributions, targeted cyber-attacks, and unauthorized intrusions that compromise critical systems [3]. Consequently, implementing robust network security mechanisms has become essential to preserve the confidentiality, integrity, and availability (CIA triad) of operational assets [4][5].

A primary challenge in securing network environments lies in the accurate detection of real-time traffic anomalies [6]. Anomaly vectors represent operational deviations from baseline normal traffic and frequently signify active network exploitations or structural breakdowns [7][8]. Manual detection methods are highly ineffective due to the massive volume of modern network exchanges and the deep complexity of underlying architectures [9][10]. As a result, the development of automated data analysis workflows capable of detecting anomalous fingerprints efficiently and with high precision has become increasingly important. Furthermore,

the development of reliable anomaly detection mechanisms contributes to Sustainable Development Goals (SDGs) 9: Industry, Innovation and Infrastructure by supporting resilient digital infrastructure, secure IoT ecosystems, and sustainable technological innovation.

Machine learning techniques have demonstrated considerable potential for automated anomaly detection due to their capacity to model complex behavioral patterns from historical data logs and perform autonomous classification [11][12]. Existing literature has largely focused on training supervised classifiers using established public repositories, such as the CICIDS2017 dataset, to recognize generalized network attacks. Although these models achieve high performance, they are constrained by their inability to capture the distinctive protocols of IoT-MQTT traffic and rarely address the data engineering challenges of live SIEM implementations.

Furthermore, prior studies have seldom addressed practical data engineering challenges encountered in live SIEM deployments, such as feature dependency and data leakage, both of which can significantly distort classification performance metrics. Existing public benchmark datasets such as CICIDS2017, while widely used, do not capture the lightweight messaging characteristics of IoT-MQTT protocols, limiting the generalizability of models trained on them to real-world IoT deployments. This study therefore pursues two primary objectives: (1) to construct a realistic IoT-MQTT benchmark dataset through a live ELK Stack deployment, and (2) to evaluate the performance of supervised machine learning classifiers on this dataset while systematically examining the effects of feature dependency and data leakage. This paper describes the dataset construction process, including the data collection pipeline, labeling procedure, and class distribution, and provides a comparative analysis of the classification performance of K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Naive Bayes algorithms.

2. METHOD

This study adopts an empirical methodology to support real-time log ingestion, feature engineering, and machine learning classification. The workflow consists of two main operational phases, namely the implementation of the Elastic Stack for data acquisition and the construction of a predictive machine learning model.

2.1. Dataset Deployment via Elastic Stack (ELK)

The data ingestion platform was developed using the ELK Stack framework (Elasticsearch, Logstash, and Kibana) to handle continuous log streams produced by IoT target devices. The acquired log records were automatically classified into normal and anomalous categories and subsequently integrated into the training dataset for analysis. The overall data collection workflow is presented in Figure 1.

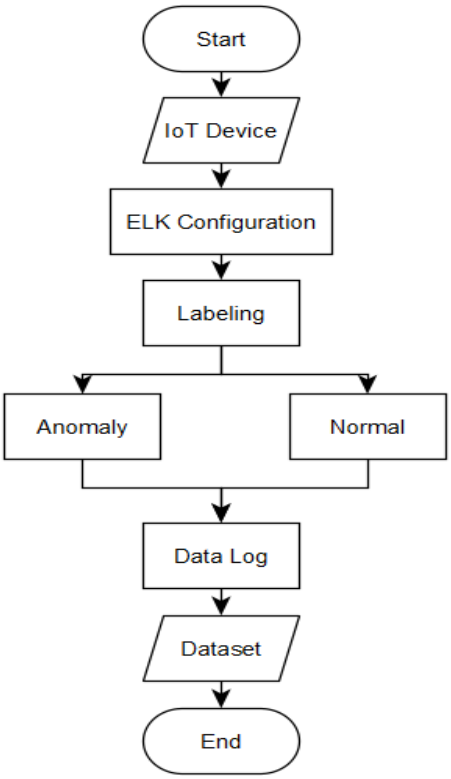

Figure 1. Flowchart of dataset collection utilizing the ELK Stack.

The proposed framework commences at the IoT device layer, where operational telemetry and environmental log data are continuously generated and transmitted. These records, comprising sensor measurements and device status information, constitute the primary data source utilized throughout this study.

During the ELK Stack deployment phase, Logstash, Elasticsearch, and Kibana were configured and integrated with an operational Eclipse Mosquitto MQTT broker to facilitate real-time data collection and monitoring. IoT devices periodically publish telemetry payloads to MQTT topics managed by the broker. Logstash subscribes to these topics, captures incoming payloads, applies parsing and transformation filters, and indexes the processed JSON records into Elasticsearch. Subsequently, Kibana is employed to visualize the ingested data streams and verify data consistency through real-time monitoring dashboards.

Following the ingestion process, an automated labeling mechanism was applied using a rule-based threshold model based on packet transmission frequency (message rate). The threshold values were derived from the observed baseline communication behavior of the deployed MQTT broker, where legitimate IoT devices consistently transmitted no more than approximately 20 messages per minute under normal operating conditions. To accommodate natural communication fluctuations and avoid false alarms, a tolerance margin of approximately 20% was applied, resulting in a normal threshold of 24 messages per minute. Traffic exceeding this baseline but remaining below 50 messages per minute was labeled as SUSPICIOUS, representing abnormal yet non-saturating behavior, whereas traffic above 50 messages per minute was classified as DDOS, as such transmission rates substantially exceeded normal MQTT communication patterns and are consistent with threshold-based intrusion detection approaches reported in previous studies[13][14]. For binary classification, both SUSPICIOUS and DDOS labels were merged into a single ANOMALY class, while NORMAL remained unchanged. The resulting dataset comprises 7,200 NORMAL records (72%) and 2,800 ANOMALY records (28%), reflecting a moderately imbalanced class distribution that is representative of realistic IoT traffic conditions where normal activity predominates.

The resulting labeled records are persistently stored as historical indices within Elasticsearch, preserving both the original system attributes and the generated security labels. Finally, these indexed datasets are exported into structured Comma-Separated Values (CSV) format to establish a master dataset for subsequent preprocessing, feature engineering, machine learning model development, and performance evaluation. This process ensures the availability of a comprehensive and reproducible dataset for anomaly detection experiments within IoT-MQTT environments.

2.2. Machine Learning Framework Design

The machine learning pipeline follows a structured six-stage research workflow: Data Collection, Data Cleaning, Feature Selection, Data Splitting, Model Training & Testing, and Performance Evaluation. The process is mapped out in Figure 2

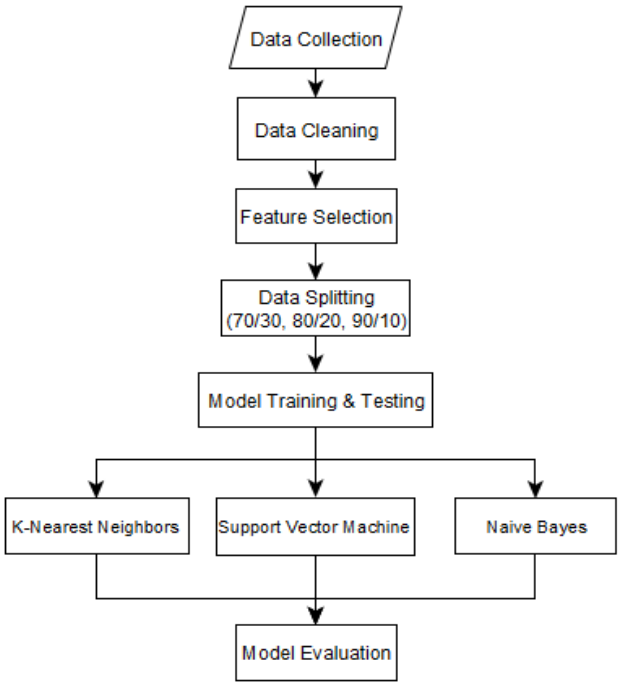

Figure 2. Research framework.

This study leverages a primary dataset of 10,000 active network transaction records compiled directly through the custom ELK Stack infrastructure [15]. Each transactional record comprises 35 distinct feature parameters, as detailed in Table 1.

Table 1. List of features from the ELK framework.

No	Feature Name	No	Feature Name
1	_attacker_ip	19	received_at
2	_attacker_ip. keyword	20	received_at. keyword
3	_id	21	rssi
4	_index	22	security_flag
5	_score	23	security_flag. keyword
6	_spoofed	24	security_status
7	_type	25	security_status. keyword
8	air_quality	26	signal_quality
9	air_quality_level	27	signal_quality. keyword
10	air_quality_level. keyword	28	source_ip
11	attack_type	29	source_ip. keyword
12	attack_type. keyword	30	date
13	device_id	31	month
14	device_id. keyword	32	year
15	device_rate	33	hour
16	ip_rate	34	minute
17	mqtt_topic	35	second
18	mqtt_topic. keyword		

Data cleaning procedures were conducted to optimize data quality before model training. Redundant attributes and duplicate feature mappings were eliminated to protect against collinearity, followed by the removal of non-contributory metadata parameters. Missing values (NaN) and null records were cleaned to prevent analytical computational failures during matrix processing.

Feature selection was completed by isolating the operational features into distinct dependent and independent vectors to focus the models on relevant parameters, as structured in Table 2. A feature dependency analysis was subsequently conducted by examining Pearson correlation coefficients between all independent variables and the target label (security_flag) to identify features with strong deterministic associations that could inflate classification performance. Features exhibiting a correlation coefficient above 0.90 with the target were flagged as potentially identity-leaking. Data leakage analysis was then performed by comparing model accuracy with and without the inclusion of the _attacker_ip feature across all three classifiers. A statistically significant drop in performance upon removal of this feature was used as evidence of leakage, indicating that the classifier was relying on an explicit identity marker rather than learning underlying behavioral patterns in the network traffic.

Table 2. Selection of variables.

No	Dependent Variable	Independent Variable
1	security_flag	_attacker_ip
2		date
3		montth
4		year
5		hour
6		minute
7		second

The cleaned dataset was partitioned into training and testing sets using three experimental split ratios (70:30, 80:20, and 90:10) to examine the influence of data allocation on classification performance. The classification stage utilized K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Naïve Bayes algorithms, which were selected due to their diverse learning mechanisms and proven effectiveness in network anomaly and intrusion detection tasks.

To validate the predictive capability of the proposed models, five standard performance metrics were employed: Accuracy, Precision, Recall, F1-Score, and Receiver Operating Characteristic (ROC) analysis. Collectively, these metrics offer a comprehensive assessment of model effectiveness by quantifying

classification accuracy, detection sensitivity, and the ability to mitigate false classifications, particularly under potentially imbalanced class distributions.

Accuracy determines the total proportion of true positive and true negative predictions relative to the entire evaluation matrix:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

Precision tracks the percentage of positive assignments that are correct against all positive instances predicted by the model:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

In other words, precision represents the proportion of correctly predicted positive instances among all instances predicted as positive by the model.

Recall measures the model's success in identifying all actual positive targets within the dataset:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

The F1-Score combines precision and recall into a balanced scalar using a harmonic mean calculation:

$$\text{F1 Score} = 2 \text{ X } \frac{\text{Precision x Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

The Receiver Operating Characteristic (ROC) curve evaluates classifier behavior by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) across varied discrimination thresholds:

$$TPR = \frac{TP}{TP + FN} \quad (5)$$

$$FPR = \frac{FP}{FP + TN} \quad (6)$$

3. RESULTS AND DISCUSSIONS

This section presents the results of the data analysis and discusses the performance of the machine learning models based on the empirical tests. The evaluation focuses on how effectively K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Naive Bayes can classify network traffic anomalies within an IoT environment using the self-collected Elastic Stack dataset.

3.1. Dataset Partitioning Performance

Following the preprocessing and data cleaning procedures, the compiled IoT-MQTT transactional log matrix was locked at a fixed total volume of 10,000 valid records. The exact row distributions for the experimental split strategies are outlined in Table 3.

Table 3. Data split distribution.

Ratio	Training Data (Rows)	Testing Data (Rows)
70:30	7.000	3.000
80:20	8.000	2.000
90:10	9.000	1.000

3.2. Algorithm Classification Metric

Table 4. Evaluation results for KNN.

Data Split Ratio	Accuracy	Precision	Recall	F1-Score
70:30	0.99	1.00	0.99	0.99
80:20	0.99	1.00	0.99	0.99
90:10	0.99	1.00	0.99	0.99

The KNN model demonstrated consistent scoring across all testing environments. In the 70:30 and 80:20 divisions, the classifier sustained an overall accuracy of 0.99, which remained consistent through the 90:10 split. A notable attribute of this distance-based classifier is its absolute precision score of 1.00 across all evaluation cycles, indicating zero false positive classifications within the network stream evaluations. The confusion matrix for the KNN classifier at the optimal 90:10 split is presented in Figure 4, confirming the near-perfect separation between the NORMAL and ANOMALY classes.

Table 5. Evaluation results for Naive Bayes.

Data Split Ratio	Accuracy	Precision	Recall	F1-Score
70:30	0.96	0.94	1.00	0.97
80:20	0.96	0.93	1.00	0.96
90:10	0.96	0.93	1.00	1.00

The Naive Bayes classifier recorded an absolute recall score of 1.00 across all training configurations. However, this was accompanied by a lower precision profile compared to the alternative models. This pattern points to a systematic bias toward positive classifications, where the model frequently classifies marginal instances as anomalies. This high recall occurs because the model aggressively flags potential threats to ensure zero false negatives, though it compromises overall accuracy due to elevated false positive counts. This behavior stems from the model's default feature independence assumption, which does not capture the functional feature correlations present in MQTT transactions. Figure 5 presents the confusion matrix of the Naive Bayes classifier at the 70:30 split, illustrating the higher rate of false positives relative to the other classifiers.

Table 6. Evaluation results for SVM.

Data Split Ratio	Accuracy	Precision	Recall	F1-Score
70:30	0.99	0.99	0.99	0.99
80:20	0.99	0.99	0.99	0.99
90:10	0.99	0.99	1.00	1.00

The Support Vector Machine reached peak classification capability at the 90:10 split ratio, achieving an accuracy score of 0.99. This performance is due to the algorithm's capability to map high-dimensional decision boundaries that maximize hyper-plane margins between target vectors. Within this IoT-MQTT environment, the SVM established clear boundaries between regular transactions and anomalies, suppressing false misclassifications relative to the probabilistic methods. The confusion matrix for the SVM classifier at the 90:10 split is shown in Figure 6, confirming the low false negative rate achieved by this classifier at its optimal configuration.

3.3. Comparative Optimization and Data Leakage Analysis

The comparative analysis shows that each classification architecture reaches its optimization point at different data partitioning ratios, as confirmed by confusion matrix analysis. The KNN and SVM models require larger training allocations (90:10) to map the underlying network complexities. Conversely, Naive Bayes reached its highest classification efficiency at the 70:30 split, showing that probabilistic models can optimize parameters with fewer training samples while utilizing wider testing matrices for structural validation. A side-by-side comparison of confusion matrices across all three classifiers at their respective best-performing split ratios is presented in Figure 7, visually illustrating the differences in false positive and false negative distributions.

Table 7. Comparison of the best results from the 3 algorithms.

Algorithm	Best Ratio	Accuracy	Precision	Recall	F1-Score
SVM	90:10	0.99	0.99	1.00	0.99
KNN	90:10	0.99	1.00	0.99	0.99
Naive Bayes	70:30	0.96	0.94	1.00	0.97

As shown in Table 7, while SVM and KNN achieved high accuracy scores of 0.99 at the 90:10 split, these metrics require critical validation due to indications of data leakage. The KNN model's absolute precision of 1.00 is largely driven by this leakage effect rather than authentic behavioral generalization. Consequently, the Naive Bayes model's accuracy of 0.96 stands out as a more scientifically valid and reliable measure of generalized performance.

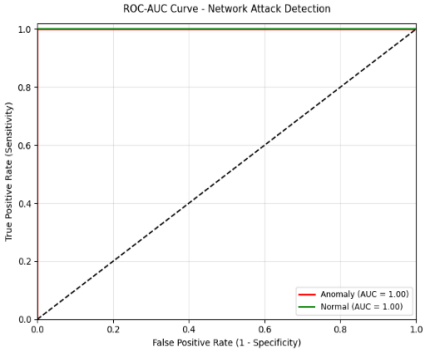

Figure 3. Combined ROC-AUC curve for the models.

As illustrated by the ROC graph in Figure 3, the classification algorithms generated identical area-under-the-curve (AUC) results of 1.00 for both target classes. This uniform performance indicates a strong reliance on the _attacker_ip feature within the independent matrix. Including the attacker's explicit IP address introduces data leakage into the training workflow; the classifiers learn to recognize specific identity fields rather than mapping the structural behavioral patterns of the underlying network anomalies. This finding underscores the necessity of filtering out direct identity fields to build models capable of generalizing to production environments.

4. CONCLUSION

This study demonstrates that K-Nearest Neighbors (KNN), Naive Bayes, and Support Vector Machine (SVM) models are capable of detecting network traffic anomalies, including DDoS and spoofing attacks, from live ELK Stack log streams in an IoT-MQTT environment. A realistic benchmark dataset was constructed through an empirical ELK Stack deployment and used to evaluate the three classifiers across multiple data partition strategies. Although SVM and KNN recorded accuracy scores of 0.99 at a 90:10 data split, these results were substantially influenced by data leakage from the _attacker_ip feature, which caused the classifiers to rely on identity markers rather than behavioral traffic patterns. Consequently, the Naive Bayes model, which achieved an accuracy of 0.96 without this dependency, represents a more reliable indicator of generalized classification performance. These findings highlight the critical importance of rigorous feature engineering and leakage detection in the development of machine learning-based intrusion detection systems. Future work should expand the dataset with more diverse attack vectors, apply feature exclusion strategies to eliminate identity-leaking attributes, test models in real-time across physical IoT edge nodes, and explore oversampling techniques alongside dimensionality reduction methods such as Principal Component Analysis (PCA) to improve robustness under imbalanced and high-dimensional conditions.

REFERENCES

[1] Makdis Nasrul, “PEMANFAATAN INTERNET UNTUK PERKULIAHAN Nasrul Makdis PENERBIT CV. PENA PERSADA,” pp. 1–20, 2020.

[2] A. H. Kelechi, U. A. Samson, M. Simeon, O. Obinna, A. Alex, and A. A. Aderemi, “The quality of service of the deployed LTE technology by mobile network operators in Abuja-Nigeria,” *Int. J. Electr. Comput. Eng.*, vol. 11, no. 3, pp. 2191–2202, 2021, doi: 10.11591/ijece.v11i3.pp2191-2202.

[3] S. W. Nourildean, M. D. Hassib, and Y. A. Mohammed, “Internet of things based wireless sensor network: a review,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 27, no. 1, pp. 246–261, 2022, doi: 10.11591/ijeecs.v27.i1.pp246-261.

[4] Tri Ginanjar Laksana and Sri Mulyani, “Pengetahuan Dasar Identifikasi Dini Deteksi Serangan Kejahatan SiberUntuk Mencegah Pembobolan Data Perusahaan,” *J. Ilm. Multidisiplin*, vol. 3, no. 1, pp. 109–122, 2024.

[5] Janet Julia Ang’udi, “Security challenges in cloud computing: A comprehensive analysis,” *World J. Adv. Eng. Technol. Sci.*, vol. 10, no. 2, pp. 155–181, 2023, doi: 10.30574/wjaets.2023.10.2.0304.

[6] A. D. Afifaturahman and F. MSN, “Perbandingan Algoritma K-Nearest Neighbour (KNN) dan Naive Bayes pada Intrusion Detection System (IDS),” *Innov. Res. Informatics*, vol. 3, no. 1, pp. 17–25, 2021, doi: 10.37058/innovatics.v3i1.2852.

[7] Abbas A. Mahdi, “Machine learning applications of network security enhancement: review,” *Comput. Sci. IT Res. J.*, vol. 5, no. 10, pp. 2283–2300, 2024, doi: 10.51594/csitrj.v5i10.1635.

[8] F. Stodt, F. Theoleyre, and C. Reich, “Advancing Network Survivability and Reliability: Integrating XAI-Enhanced Autoencoders and LDA for Effective Detection of Unknown Attacks,” *20th Int. Conf. Des. Reliab. Commun. Networks, DRCN 2024*, pp. 9–16, 2024, doi: 10.1109/DRCN60692.2024.10539141.

[9] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, “Machine Learning for Anomaly Detection: A Systematic Review,” 2021. doi: 10.1109/ACCESS.2021.3083060.

[10] M. D. Ogah, J. Essien, M. Ogharandukun, and M. Abdullahi, "Machine Learning Models for Heterogenous Network Security Anomaly Detection," *J. Comput. Commun.*, vol. 12, no. 06, pp. 38–58, 2024, doi: 10.4236/jcc.2024.126004.
[11] M. Mujiono, D. A. Larasati, M. Hemansyah, and F. Fatimatuzzahra, "Deteksi Anomali dalam Sistem Keamanan Jaringan Menggunakan Teknik Supervised Machine Learning," *J. Esensi Infokom J. Esensi Sist. Inf. dan Sist. Komput.*, vol. 9, no. 1, pp. 65–69, 2025, doi: 10.55886/infokom.v9i1.971.
[12] A. S. M. Kayes, W. Rahayu, T. Dillon, A. Salehi Shahraki, and H. Alavizadeh, "Safeguarding Individuals and Organizations From Privacy Breaches: A Comprehensive Review of Problem Domains, Solution Strategies, and Prospective Research Directions," *IEEE Internet Things J.*, vol. 12, no. 2, pp. 1247–1265, 2025, doi: 10.1109/JIOT.2024.3481316.
[13] K. M. D. Pertiwi, N. A. Hasibuan, and D. P. Rahmawati, "Denial of Service (DOS) Attack Detection on MQTT Protocol Using the Random Forest Method," *J. Online Inform.*, vol. 11, no. 1, pp. 48–59, 2026, doi: 10.15575/join.v11i1.1784.
[14] D. B. Negesse, K. A. Gemeda, and G. Gianini, "DDoS attack detection and classification for the MQTT-IoT protocol using LSTM models," *Discov. Appl. Sci.*, vol. 8, no. 5, 2026, doi: 10.1007/s42452-026-08563-8.
[15] R. R. Billanivo, "IoT-MQTT ELK Benchmark Dataset," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/rrbillanivo/iot-mqtt-elk-benchmark-dataset