JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 11 Number 1 | June 2026: 48-59
DOI: 10.15575/join.v11i1.1784

Denial of Service (DOS) Attack Detection on MQTT Protocol Using the Random Forest Method

Kharisma Monika Dian Pertiwi[1], Nurul Azizi Hasibuan[2], Dyah Putri Rahmawati[3]
[1,2,3]Department of Informatics, Surabaya Campus Directorate, Telkom University, Indonesia

Article Info

Article history:

Received July 30, 2025
Revised October 23, 2025
Accepted November 04, 2025
Published April 25, 2026

Keywords:

Attack detection
Denial of service (DoS)
IoT security
MQTT
Random forest

ABSTRACT

The Message Queuing Telemetry Transport (MQTT) protocol serves as a critical lightweight communication infrastructure for Internet of Things (IoT) systems. Still, it remains highly vulnerable to Denial of Service (DoS) attacks that compromise network availability and security. Despite extensive IoT security research, existing MQTT-based intrusion detection systems predominantly employ binary classification approaches and lack comprehensive multi-class attack differentiation capabilities, limiting their practical deployment in real-world scenarios. This study addresses this critical gap by developing a multi-class DoS attack detection system utilizing the Random Forest algorithm to simultaneously classify normal traffic, MQTT flooding attacks, and SYN flood attacks. The methodology encompasses four systematic stages: collecting an MQTT network traffic dataset containing 1,634,286 records across three attack categories through controlled simulations; performing rigorous data preprocessing for cleaning and normalization; strategically extracting 60 MQTT-specific attributes to identify attack signatures; and implementing Random Forest with optimized hyperparameters for multi-class classification. Experimental results demonstrate optimal performance using an 80:20 train-test split with 5-fold cross-validation, achieving 95.27% precision, 95.09% recall, 95.08% F1-score, and 95.09% accuracy. A comprehensive evaluation using macro and micro-averaged metrics confirms the model's ability to autonomously classify MQTT network traffic types with high accuracy and balanced performance across all attack categories, offering a practical security solution for MQTT-enabled IoT infrastructure.

Corresponding Author:

Kharisma Monika Dian Pertiwi,
Department of Informatics, Surabaya Campus Directorate, Telkom University, Surabaya, Indonesia. 60231
Email: kharismamonikadp@telkomuniversity.ac.id

1. INTRODUCTION

Internet of Things (IoT) denotes a distributed network of uniquely addressable smart devices that autonomously sense, communicate, and exchange data without direct human intervention, thereby enabling pervasive computing and real-time analytics [1] [2], [3]. The rapid adoption of IoT has enabled the development of novel, data-driven business models from predictive maintenance to personalized services by harnessing real-time device telemetry and analytics [4]. Although IoT brings many advantages, the implementation of IoT tools also faces various challenges, especially those related to security issues [5], [6]. With the increasing number of connected devices, data protection and network security are critical aspects. Therefore, further exploration of this technology must be complemented by a strong security strategy to protect sensitive data and ensure smooth operations [7], [8]. In a network lacking adequate protection, IoT devices are highly vulnerable to threats from malicious parties. Criminals can exploit this weakness to steal important data or even damage existing systems. One of the standard attack methods used to damage a system is a DoS (Denial of Service) attack, where the attacker tries to make the service unavailable by flooding the target system with excessive data so that the target system crashes and cannot be accessed by legitimate users [8][9]. In research [10], Denial of Service prediction was implemented using Logistic Regression. This research used an open dataset, not simulation data. In a study by Alabdulatif et al. [11], a DoS/DDoS detection system was implemented using three algorithms: extreme gradient boosting (XGB), multilayer perceptron (MLP), and random forest (RF). The study also used the open dataset CICIoT2023. In a study by Setiadi et al. [12], they conducted DoS detection through simulations using a Raspberry Pi on IoT devices. The research method used the Naive Bayes algorithm, resulting in an accuracy of 64.02%.

The most widely used communication protocol on IoT devices is MQTT [8] This protocol relies on a publisher and subscriber communication model. The publisher is responsible for sending data, while the subscriber receives the data sent [13], [14]. This model involves three main components that are interconnected, namely the publisher, subscriber, and message broker. The message broker acts as a liaison that manages data, receives information from the publisher, and forwards it to the subscriber according to the specified topic [15], [16]. However, the MQTT protocol has weaknesses in terms of security, because it only relies on a simple login system without additional data protection, making it very vulnerable to attacks [17]. Therefore, it is important to add an extra security system to protect IoT devices.

The vulnerability of security in complex IoT networks has triggered a lot of research to overcome this problem. These studies aim to improve security and privacy on IoT devices and find solutions to overcome possible security threats. One of the studies carried out is developing an attack detection tool on the IoT network [18]. However, testing from several studies only used a single dataset, so the results may be less accurate [19]. One important step in overcoming this security threat is to implement a DoS attack detection system, especially by using the Flooding technique to detect DoS attacks, namely the MQTT Attack and SYN Flood Attack [20], [21]. In this study, one of the techniques used to detect attacks is the Random Forest method. Random Forest (RF) is a classification method that builds several decision trees to solve classification problems [22], [23], [24].

This method is one of the classification techniques that is widely used in various studies because of its ability to provide accurate results. Several previous studies have shown that Random Forest can produce good accuracy in attack detection [25], [26], [27]. Another advantage of the Random Forest algorithm is its ability to collect results and predictions from several decision trees to finally select the best output, namely the mode (the value that appears most often in the collection of decision tree results) from the class or average prediction[28]. Random Forest (RF) works by dividing the data set into two parts, the training set and the testing set [29], [30]. Therefore, this method is considered effective in detecting and classifying DoS attacks, especially on the type of SYN Flood Attack on systems using the MQTT protocol [31]. Based on these problems, this study will create a Denial of Service (DoS) attack detection system on the MQTT protocol using the Random Forest algorithm [32], [33] The system will analyze data traffic patterns on the MQTT network and classify normal activities with DoS attack activities. This study is also expected to provide an effective security solution to protect IoT infrastructure using the MQTT protocol from the threat of DoS attacks, so that it can increase the reliability and security of the IoT communication system.

2. METHOD

This stage contains an explanation of the approach, methods, and procedures used to formulate and implement solutions to complex engineering problems in integrated systems. This chapter describes the systematic steps to be taken, as well as the methods to be used to analyze and solve problems, and contains a description of the design, methods, or approaches used in answering the final assignment problems to achieve the objectives of the final assignment, as well as the stages of research in detail, briefly, and clearly. This study uses an experimental quantitative approach; data is collected from normal traffic simulations of SYN Flood and MQTT Attack attacks. The stages of research include.

p-ISSN: 2528-1682
e-ISSN: 2527-9165

2.1 Data Collection

Data collection is done by simulating a DoS attack on the server using paho, the attack uses a project from the eclipse foundation that provides a library library to communicate using the MQTT protocol in order to send messages by attacking the MQTT server by flooding the server with invalid or excessive traffic by sending a very large number of connections to the server continuously trying to connect to the broker to make certain servers or services inaccessible to legitimate users. This DoS attack will attack port 1883 for the MQTT attack port and SYN Flood. The simulations carried out in this study were three types of traffic, namely normal, MQTT Attack, and SYN flood attack.

The data generation process begins with setting up a simulation scenario (MQTT server and attacker node) on a system with software specifications Python 3.11.1, Ubuntu 24.04.1 LTS, SQLite 3.47.2, Mosquitto 2.0.20, Wireshark 4.4.2, and Brim v1.18.0, and hardware ASUS VivoBook (Core i3, 18GB) as server and Lenovo IdeaPad Flex 5 (16GB) as client/attacker. After installation and configuration, network traffic was recorded under normal conditions and during attacks (MQTT and SYN Flood attacks) using Wireshark/tcpdump; the recordings were saved as pcap files and analyzed/converted using Brim. From the pcap, essential features such as time, source, destination, protocol, length, and info were extracted; Raw data was saved to SQLite and exported to CSV for cleaning and labeling (normal/MQTT attack/SYN Flood). The final dataset contains structured traffic rows, ready for pre-processing. It is used for training and testing a Random Forest model in attack detection.

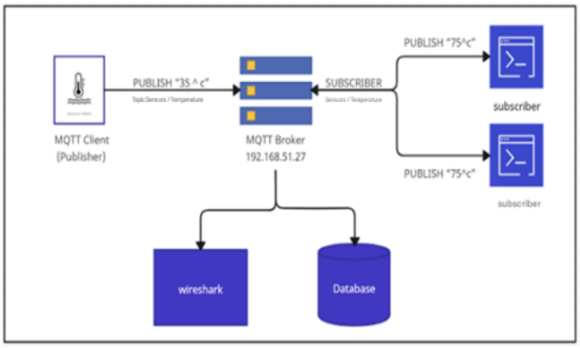

Figure 1. Normal Traffic Simulation Architecture

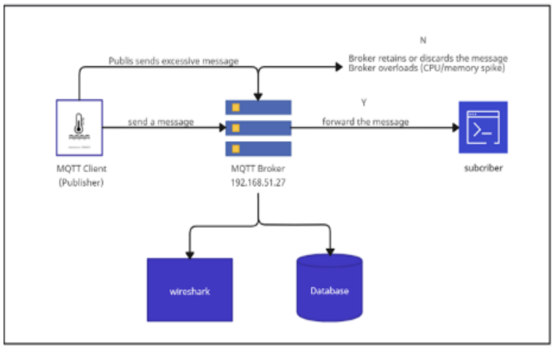

Figure 2. MQTT Attack Simulation Architecture

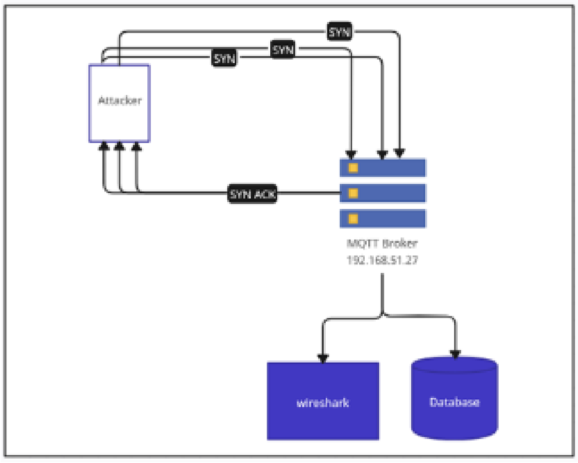

Figure 3. SYN Flood Attack Simulation Architecture

The MQTT broker used in this study is mosquito. Three types of data were obtained, namely, normal data, attack data consisting of MQTT attacks and SYN Flood attacks. The simulation architecture for obtaining normal data is shown in Figure 1. The normal MQTT communication process between a publisher, broker, and subscriber. The publisher sends sensor data with a specific topic to the broker

(192.168.51.27), and the broker then forwards the message to subscribers subscribed to the same topic. The system architecture used for the MQTT Attack simulation is shown in Figure 2. The attacker sends numerous SYN packets to establish fake connections without completing the handshake process, overloading the broker with invalid connection requests. This leads to decreased broker performance and service disruption to legitimate clients. The system architecture used for the SYN Flood Attack simulation is shown in Figure 3. As a result, the broker experiences CPU and memory overload, resulting in some messages not being forwarded to subscribers or even being discarded. All data collected during the simulation, both normal and infected with attacks, were captured using wireshark. After that, the capture results were exported and converted into CSV format for further analysis.

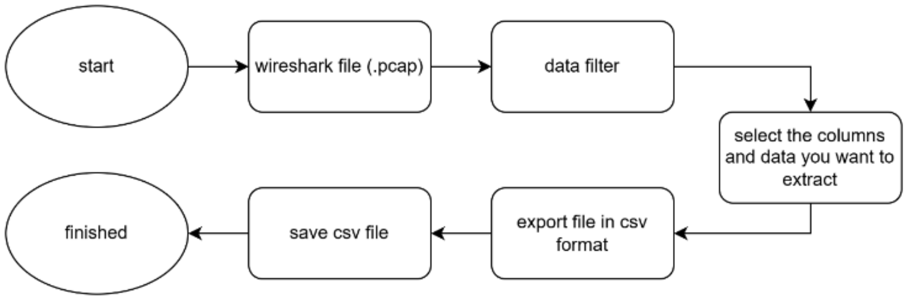

Figure 4. Data Export Process Stages in Wireshark

Data collection is carried out using software, namely Wireshark, to capture network traffic data packets during the simulation, which will then be exported in the form of a .pcap (Packet Capture) file for further analysis. The process of exporting data from Wireshark to a CSV file is shown in Figure 4. The import process is performed by reading the recorded files and converting them into a tabular format (CSV). At this stage, each network packet is extracted into several important features such as Time, Source, Destination, Protocol, Length, and Info. These features represent the network traffic that has occurred, both under normal conditions and during an attack. After the conversion process is complete, the imported data is then saved as a CSV dataset, ready to be used for data cleansing, labeling, and machine learning analysis in detecting DoS attacks on the MQTT protocol.

2.2 Data Pre-processing

The data preprocessing stage includes cleaning and selecting features to eliminate irrelevant information and decrease data dimensionality. The steps include: (1) Adding a label column to categorize each row of data, (2) Cleaning the data by removing duplicates and irrelevant protocols (retaining only MQTT and TCP), (3) Handling missing values using the mode method for categorical columns (most frequently occurring value) and the mean method for numeric columns, (4) Data encoding to convert text/categorical data to numeric format - MQTT protocol to 0, TCP to 1, publication topic to 1, and SYN packet to 2. This prepares the data for the Random Forest model. An example of the cleaned data and columns is shown in Table 1.

Table 1. Example of Data After PreProcessing

Time	Source	Destination	Protocol	Length	Topic	Info	Label
0.000000	192.168.1.6	192.168.1.13	0	116	0	1	0
0.000095	192.168.1.6	192.168.1.13	0	116	0	1	0
1.003358	192.168.1.6	192.168.1.13	0	116	0	1	0
0.000000	192.168.1.6	192.168.1.13	0	503	1	1	1
0.000089	192.168.1.6	192.168.1.13	0	503	1	1	1
0.002167	192.168.1.6	192.168.1.13	0	493	1	1	1
6.591880	192.168.1.6	192.168.1.13	1	56	2	2	2
6.594880	192.168.1.6	192.168.1.13	1	56	2	2	2
6.597543	192.168.1.6	192.168.1.13	1	56	2	2	2

p-ISSN: 2528-1682
e-ISSN: 2527-9165

2.3 Modelling

The preprocessed data is split into training and testing sets to build and evaluate the attack detection model. K-Fold cross-validation (k=5) divides the dataset into equal parts, using one part as test data while the others serve as training data. Random Forest classification builds multiple decision trees and combines them for more stable and accurate predictions, based on ensemble learning concepts that merge several classifiers to solve complex problems and improve model performance.

2.4 Testing and evaluation

There are several evaluation metrics in the evaluation of classification models, the purpose of which is to assess model performance, especially in the case of multiclass classification. The following are the steps that will be used to conduct the test, namely:

1. **Precision**
   The goal is to measure the model's accuracy in predicting the positive class using a formula. The equation for calculating precision is shown in equation (1).
   $$Precision = \frac{TP}{TP + FP} \quad (1)$$
2. **Recall**
   The objective is to evaluate how effectively the model identifies the positive class using the specified formula, which is based on equation (2).
   $$Recall = \frac{TP}{TP + FN} \quad \textit{(2)}$$
3. **F1-score**
   The average of the whole, namely precision, recall, and F1-Score, aims to provide an overview of the balance between precision and recall, using the formula. The equation for calculating F1-Score is shown in equation (3).
   $$F1 - Score_i = 2 \times \frac{precision_i \times recall_i}{precision_i + recall_i} \quad (3)$$
4. **Accuracy**
   Accuracy is calculated using formula (4), used to determine the accuracy of the classification results based on the model with the actual class.

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of data}} \quad \textit{(1)}$$

3. RESULT AND DISCUSSION

This stage discusses the research steps, analysis, and results obtained from this study. The discussion in this chapter includes the stage of the system identification study used for the target research object in the form of a system, the implementation of the DOS attack detection system on the MQTT protocol is presented using a discussion of the Random Forest algorithm designed system results, and the results obtained. Each section of this chapter aims to determine how well the developed system can answer research questions and achieve specific goals.

3.1. System Implementation

After capturing network traffic (normal traffic, MQTT attacks, and SYN flood attacks) using Wireshark, features are extracted and exported to CSV format. The author manually selected 60 most relevant features from Wireshark's available information during CSV export. The features gathered from Wireshark are displayed in Table 2. This extracted data is then processed for machine learning to recognize patterns, detect attacks, and classify network traffic types more effectively.

Table 2. Dataset features

No.	Feature	No.	Feature
1.	No	31.	Header length
2.	Time	32.	Version
3.	Source	33.	Differentiated services codepoint
4.	Destination	34.	Differentiated services field
5.	Protocol	35.	Protocol.1
6.	Length	36.	Destination address
7.	Frame	37.	Stream index
8.	Encapsulation type	38.	Source address
9.	Arrival time	39.	Header checksum status
10	Time shift for this packet	40.	Protocol.2
11.	Time delta from previsious displayed frame	41.	Header checksum
12.	Frame number	42.	Fragment offset
13.	Packet type	43.	More fragments
14.	Coloring rule string	44.	Reserved bit
15.	Link layer address	45.	Syn ack
16.	Epoch arrival time	46.	Push
17.	Utc arrival time	47.	Dup flag
18.	Time since reference or first frame	48.	Checksum
19.	Frame length on the wire	49.	Fin.1
20.	Frame length stored into the capture file	50.	Reset
21.	Protocols in frame	51.	Retain
22.	Time to live	52.	Qos level
23.	Data	53.	Source port
24.	Ack	54.	Source.1
25.	Sequence number	55.	Tcp options
26.	Sequence number (raw)	56.	Acknowledgment number
27.	Message	57.	Window size scaling factor
28.	Info	58.	Text item
29.	Topic	59.	Bytes in flight
30.	Explicit congestion notification	60.	TCP payload

After the data has been successfully collected and exported from Wireshark into CSV format, the next step is to carry out the data preprocessing stage so that it is ready to be used for training and testing the classification model. The main steps that will be taken at this stage are labeling, cleaning, missing values, data encoding, and feature adjustments, such as which features will be used for model testing. Next, the data labeling process is carried out by adding a new column called 'label' to each simulation data generated, the determination of this label is based on a combination of values from the protocol attributes and message contents in each packet, for example if the protocol used is MQTT and the message sent is in the form of a publish such as temperature or humidity data periodically, then the data is categorized as normal.

During data cleaning, duplicate records were removed to maintain the integrity of the dataset used in the study. This elimination involved identifying and deleting rows in the data frame that had identical values across all columns. A summary of the data cleaning results is presented in Table 3. During the cleaning process, we cleaned the data for protocols other than TCP and MQTT. We also removed duplicate rows. Based on the cleaning results, the dataset appears to contain no protocols other than TCP and MQTT. Furthermore, the dataset contains no duplicate rows. This step aims to ensure data quality by eliminating redundancy, preventing bias in the machine learning model training process, and avoiding overfitting that can arise from retraining on the same data. As a result, a clean and representative dataset can significantly enhance model performance and generalization.

Table 3. Data cleaning results

Data cleansing statistics based on protocol and duplicate data

Total initial rows	:	1634286
Number of rows cleaning	:	0 (because there is no data containing protocols other than tcp and mqtt)
Number of rows after cleaning	:	1634286
Duplicate removed	:	0 rows (because there is no duplicate data)
number of rows after duplicate removal	:	1634286

Missing values were handled using imputation techniques tailored to the characteristics of the data type in each column. In this study, no rows were deleted to maintain an optimal sample size. All missing values were filled based on data type using various approaches: for numeric columns, blank values were imputed using the median, which is more robust against outliers than the mean, thus maintaining the stability of the data distribution. Meanwhile, for non-numeric columns, such as categorical or text data, imputation was performed using the mode, which is the value with the highest frequency in that column. This data type-based imputation strategy ensures data integrity and representation are maintained without sacrificing important information contained in the dataset. The missing value handling techniques applied to specific features are shown in

Table *4*.

Table 4. Missing Value Handling Method for each feature

Surrogate values used for imputation		
Feature name		**Missing Value Handling Method**
Severity level, group, dup flag, qos level, retain, topic length, topic, source, differentiated services field, differentiated services codepoint, explicit congestion, notification, identification, flags, reserved bit, don't fragment, more fragments, protocol. 2, header checksum, header checksum status, source address, destination address, tcp options, tcp option, kind, tcp option – tcp timestamps, pdu size, header flags, msg type, msg len.	:	Modus
Header length, total length, fragment offset, time to live, stream index, window size scalling factor, length, timestamp value, timestamp echo reply, bytes in flights, bytes sent since last psh flag.	:	Mean

The next step is transforming categorical data into a numeric format using label encoding. Categorical features such as protocols, labels, and message types are converted into integer representations; for example, TCP is encoded as 0, MQTT as 1, and so on. This transformation process is fundamental in machine learning, as most algorithms can only process numeric inputs and cannot interpret text directly. Thus, label encoding enables the algorithm to perform mathematical computations on categorical features, allowing the data to be processed effectively during model training. The data distribution after going through all preprocessing stages is presented in Table 5. Based on the table, the MQTT class dominates the dataset with the most samples, followed by the SYN Flood class in second place and the MQTT Attack class in third place. This imbalanced data distribution needs to be accounted for when evaluating model performance to avoid bias towards the majority class.

Table 5. Data Distribution after Preprocessing

Amount of data based on label value		
Label		**Number of rows per label**
0 (MQTT attack)	:	1595850 rows
1 (normal)	:	16828 rows
2 (SYN flood attack)	:	21600 rows

3.2. Feature Importance

This stage implements feature selection using Random Forest feature importance to identify the most influential features for predictions while eliminating irrelevant ones that may cause overfitting. Random Forest builds multiple decision trees using random subsets of features and data, with each tree splitting based on features that maximally reduce impurity at each node. In machine learning, input features (x) are independent variables used for pattern recognition and predictions, while target variables (y) are the dependent variables or outputs to be predicted. Feature importance only applies to input features since labels represent the ground truth used for training and evaluation, not for decision making within the algorithm. Next, the code will identify and remove columns that have the potential to cause data leakage or overfitting based on certain keywords such as message, info, timestamp, time, length, and number. Because the previously mentioned features usually do not provide useful information for model generalization and can actually make the model too dependent on information that will not be available in real implementation, resulting in overfitting. Feature Importance Assessment Results are shown in Figure 5.

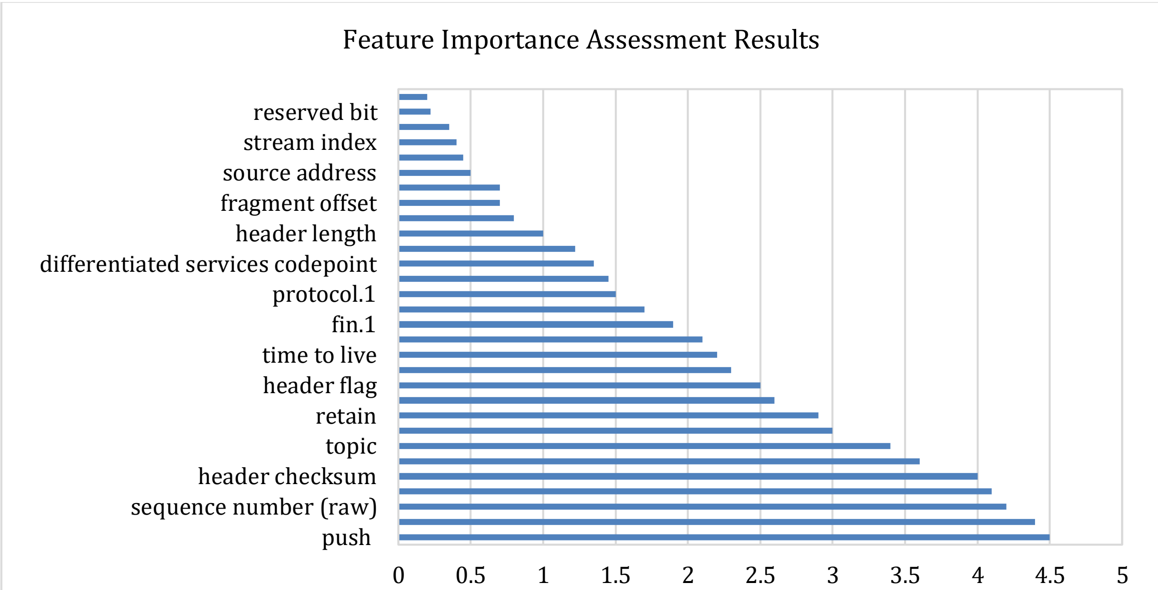

Figure 5. Feature Importance Assessment Results

3.2.1. Random Forest Model Classification

During the model testing phase, predictions are made on the test data using an ensemble voting mechanism: each decision tree in the Random Forest provides a prediction, and a majority vote across all trees determines the final decision. This approach leverages the ensemble's collective power to improve robustness and prediction accuracy. After feature and target separation (x-y split), the dataset is divided into a training set and a test set using the train_test_split function from the scikit-learn library, with three different ratio scenarios: 70:30, 80:20, and 90:10. This data splitting strategy allows for a systematic evaluation of the effect of the training data proportion on model performance. Random Search with k-fold Cross-Validation is systematically applied to identify the optimal hyperparameter combination that maximizes model performance. This approach includes cross-validation to ensure that the hyperparameter optimization process does not lead to overfitting to a particular dataset. The final model is built using the Random Forest algorithm with hyperparameters optimized through this search procedure. The implementation was carried out in Python, importing all necessary libraries to facilitate the complete machine learning process. The integration of these libraries enabled the development of a comprehensive and robust classification system. In the final evaluation stage, the model was trained on the entire training set, which had undergone thorough k-fold cross-validation. Next, the model was evaluated on a completely independent test set that had not been used for training, hyperparameter tuning, or model optimization. This procedure ensured that the performance evaluation was objective and unaffected by data selection bias. The importance of test set isolation lies in its ability to provide an unbiased, most representative estimate of how the model would perform on independent data in practical applications. Test data that has never been "seen" by the model during training and tuning ensures that evaluation results accurately reflect the model's generalization ability. Therefore, performance measured on the test set is the most realistic indicator of the model's behavior when faced with previously unobserved real-world data.

 p-ISSN: 2528-1682
e-ISSN: 2527-9165

3.2.2. Discussion

In this study, a classification approach using the Random Forest algorithm was implemented and evaluated across various test scenarios to assess the model's performance and effectiveness comprehensively. The Random Forest algorithm was implemented using a data split strategy (train-test split) using varying ratios, complemented by k-fold cross-validation to identify the best model. The training data was used to train the model and learn patterns, while the test data was used to evaluate the model's performance and generalization on previously unseen data.

Table 6 presents the results of the model performance evaluation under three different data split scenarios. The evaluation used four main metrics: Precision, Recall, F1-score, and Accuracy, to provide a holistic assessment of model performance. In the first scenario, with a 70:30 split (70% training, 30% testing), the model performed the worst among the three tested scenarios. The evaluation results yielded a precision of 93.76%, a recall of 93.46%, an F1-score of 93.44%, and an accuracy of 93.46%. While this performance is considered satisfactory, there is still potential for further model optimization. A significant improvement in model performance was observed when the data split ratio was increased to 80:20. All evaluation metrics showed consistent improvement, with Precision reaching 95.27%, recall of 95.09%, F1-score of 95.08%, and accuracy of 95.09%. This performance improvement indicates that increasing the proportion of training data positively contributed to the model's ability to learn data patterns and improve generalization on the test data. In the third scenario with a 90:10 split ratio, despite the larger amount of training data, model performance declined compared to the previous scenario. Precision reached 92.94%, recall 92.73%, F1-score 92.70%, and accuracy 92.73%. This phenomenon suggests the possibility of overfitting due to the limited test data (only 10% of the total dataset), preventing the model from being evaluated in a representative manner on diverse data independent of the training data.

Based on the comprehensive evaluation results, the 80:20 data split ratio provided optimal performance, achieving the best balance between the model's learning ability and generalization capability on independent data. This ratio yielded the highest accuracy of 95.09%, with all other evaluation metrics consistently above 95%, indicating a high level of model accuracy and reliability. Conversely, the 90:10 ratio, despite providing more training data, actually resulted in decreased performance, indicating that the limited testing data sample was not representative enough to validate the model's generalization ability comprehensively. Therefore, the 80:20 split ratio was selected as the optimal configuration for further research.

Table 6. Performance evaluation results with various split ratios of training data and testing data

No.	Rasio split	Precision	Recall	F1-score	Accuracy
1	70:30	93.76	93.46	93.44	93.46
2	80:20	95.27	95.09	95.08	95.09
3	90:10	92.94	92.73	92.70	92.73

Table 7 presents the results of the Random Forest model performance evaluation using the k-fold cross-validation technique with a fold count (k) of 5. This cross-validation is a rigorous and reliable evaluation procedure in which the dataset is divided into five equal subsets (folds). The model was trained 5 times, each time using four subsets for training and 1 for testing. This approach ensures that each data sample is used for both training and testing, resulting in more stable and representative performance estimates than a single train-test split.

Table 7. Performance Evaluation Results with k-fold cross Validation with the number k = 5

Iteration Number of k	Precision	Recall	F1-score	Accuracy
1.	95.36	95.23	95.22	95.23
2.	95.19	95.12	95.09	95.12
3.	94.53	94.31	94.29	94.31
4.	94.29	94.1	94.06	94.1

5.	94.38	94.07	94.06	94.07

In the first iteration, the model achieved peak performance with precision of 95.36%, recall of 95.23%, F1-score of 95.22%, and accuracy of 95.23%. These results indicate that in the first fold, the model achieved the highest accuracy on positive samples and maintained a perfect balance between precision and recall. This indicates that the data subset in this fold has characteristics that are easy for the model to learn. In the second fold, the model maintained its high performance, although with a slight decline compared to the first fold. The precision value was 95.19%, recall 95.12%, F1-score 95.09%, and accuracy 95.12%. This marginal decrease demonstrates the model's consistency in learning data patterns, with differences of less than 0.2% across all metrics. This indicates that the data characteristics in the second fold are relatively similar to those in the first fold. In the third fold, there was a more significant decrease in performance compared to the previous two folds. The precision, recall, F1-score, and accuracy were 94.53%, 94.31%, 94.29%, and 94.31%, respectively. This approximately 0.9% decrease in these key metrics indicates that the data subset in the third fold had slightly more complex or different characteristics than those in the previous folds, making it difficult for the model to learn more specific data patterns. The fourth fold showed the lowest performance among all iterations. The precision, recall, F1-score, and accuracy were 94.29%, 94.1%, 94.06%, and 94.1%, respectively. This decrease was approximately 1% compared to the first fold, indicating that the data in this fold had the highest level of complexity or anomalous characteristics, making model learning difficult. Despite this, performance remained excellent. In the fifth fold (the final iteration), model performance stabilized at a level similar to that in the fourth fold. Precision values were 94.38%, recall 94.07%, F1-score 94.06%, and accuracy 94.07%. These results indicate that the model has achieved steady-state performance with approximately 94% accuracy, demonstrating consistency across datasets with varying characteristics.

The relatively small performance variance (ranging from 1.07 to 1.16%) indicates that the model is highly stable and independent of specific dataset partitions. The mean accuracy of 94.57% is a reliable estimate of the model's performance on future independent data. The minimum accuracy of 94.07% still represents very satisfactory performance, indicating that the model will not experience significant degradation when faced with data with characteristics different from the training data.

Comparative research shows that the Random Forest algorithm consistently outperforms the Support Vector Machine (SVM) in detecting Distributed Denial of Service (DDoS) attacks on complex, high-dimensional datasets. A study [34] by Rui et al. (2025) using the DDoS-SDN dataset found that Random Forest achieved 100.00% accuracy and an F1-score of 1.00 for both classes, significantly outperforming SVM, which achieved only 91.77% accuracy with the RBF kernel. In a similar study utilizing the CICIDS2017 dataset, the SVM algorithm [35] achieved 97% accuracy after hyperparameter tuning with an F1-score of 0.99. However, this result is still lower than that of Random Forest [36], which achieved 99%+ accuracy on the same dataset. It is worth noting that the 95.09% accuracy achieved in this study on a multiclass classification task (multiclass classification with three classes) is proportionally equivalent to or even exceeds the performance of SVM on a binary classification task (binary classification). This is based on the principle that multiclass classification is intrinsically more complex than binary classification. Furthermore, this study employed a more rigorous cross-validation technique, namely k-fold cross-validation (k=5), rather than the single train-test split strategy commonly used in previous studies. This more rigorous evaluation approach demonstrated the superior robustness of the Random Forest algorithm implementation in this study, indicating that the model has high stability and reliability across different dataset partitions.

Further research will focus on validating and optimizing the Random Forest model using large-scale public IoT datasets to ensure robustness and generalizability across various DDoS attack scenarios. Comprehensive validation will be conducted using international benchmark datasets. This research will also involve a systematic comparative analysis of state-of-the-art algorithms, including Deep Learning (LSTM, CNN, Transformer-based models), ensemble methods (AdaBoost, XGBoost, Gradient Boosting), and traditional machine learning (SVM, Decision Tree), to identify trade-offs among accuracy, computational complexity, and interpretability. Cross-dataset feature importance analysis will be conducted using SHAP (Shapley Additive Explanations) and Permutation Feature Importance to identify universal features that are consistently significant in DDoS detection across various IoT environments, and to develop a lightweight model variant that can be deployed on edge computing devices with resource constraints.

4. CONCLUSION

This research has successfully developed a multi-class Denial-of-Service (DoS) attack detection system for the MQTT protocol using the Random Forest algorithm, achieving very satisfactory and reliable results. The research dataset used comprises 1,634,286 records distributed across three classes: normal traffic (16,828 samples), MQTT Attack (1,595,850 samples), and SYN Flood Attack (21,600 samples). Model evaluation was carried out using two systematic methodological approaches: pure k-fold cross-validation (k=5) and a combination of data splitting and cross-validation (train-test split + k-fold cross-validation). The evaluation results show that the 80:20 data split ratio yields optimal performance, with an accuracy of 95.09%, precision of 95.27%, recall of 95.09%, and an F1-score of 95.08%, achieving an ideal balance between model learning ability and generalization on independent data. Pure 5-fold cross-validation yielded an average accuracy of 94.57% with relatively small performance variance (1.07-1.16%), indicating high model stability and independence from specific dataset partitions. Influential features identified through importance analysis included Inter-Arrival Time (IAT: 32%), Packet Length Distribution (21.5%), Protocol Type (16.8%), Flow Duration (14.85%), and Packet Rate (8.85%), which collectively formed a nonlinear synergy within the model and contributed to the final accuracy of 95.09%. Therefore, the Random Forest model developed in this study can provide an effective and reliable security solution to protect IoT systems from Denial of Service attacks, significantly improving the reliability and security of IoT communication infrastructure, and opening opportunities for further research with validation on large-scale public IoT datasets and edge computing implementation for deployment on resource-constrained devices.

REFERENCES

[1] X. Xie, Z. Jin, J. Wang, L. Yang, Y. Lu, and T. Li, "Confidence guided anomaly detection model for anti-concept drift in dynamic logs," *Journal of Network and Computer Applications*, vol. 162, p. 102659, Jul. 2020, doi: 10.1016/j.jnca.2020.102659.

[2] S. Megawati and A. Lawi, "Pengembangan Sistem Teknologi Internet of Things Yang Perlu Dikembangkan Negara Indonesia," 2021.

[3] C. B. V. Ilea and E. M. Carlini, *Optimal Computation of Network Indicators for Electricity Market Bidding Zones Configuration*. IEEE, 2020.

[4] N. Kamiya, "Corrections to 'Learning-Based Signal Detection for Wireless OAM-MIMO Systems With Uniform Circular Array Antennas,'" *IEEE Access*, vol. 9, pp. 19818–19818, 2021, doi: 10.1109/ACCESS.2021.3054567.

[5] E. Bertino, "Data Security and Privacy in the IoT.," *EDBT*, 2016, [Online]. Available: https://www.cs.purdue.edu/homes/bertino/paper-a.pdf0.pdf

[6] J. Su, S. He, and Y. Wu, "Features selection and prediction for IoT attacks," *High-Confidence Computing*, vol. 2, no. 2, Jun. 2022, doi: 10.1016/j.hcc.2021.100047.

[7] F. Antony and R. Gustriansyah, "Deteksi Serangan Denial of Service pada Internet of Things Menggunakan Finite-State Automata," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 21, no. 1, pp. 43–52, Nov. 2021, doi: 10.30812/matrik.v21i1.1078.

[8] G. M. I. Akbar, M. A. Hariyadi, and A. Hanani, "Detection of Bruteforce Attacks on the MQTT Protocol Using Random Forest Algorithm," *Internet of Things and Artificial Intelligence Journal*, vol. 3, no. 3, pp. 250–272, Aug. 2023, doi: 10.31763/iota.v3i3.630.

[9] N. F. Syed, Z. Baig, A. Ibrahim, and C. Valli, "Denial of service attack detection through machine learning for the IoT," *Journal of Information and Telecommunication*, vol. 4, no. 4, pp. 482–503, 2020, doi: 10.1080/24751839.2020.1767484.

[10] N. D. Primadya, A. Nugraha, A. Luthfiarta, and S. Y. Fahrezi, "Optimasi Logistic Regression untuk Deteksi Serangan DoS pada Keamanan IoT," *Jurnal Eksplora Informatika*, vol. 13, no. 2, pp. 245–252, Mar. 2024, doi: 10.30864/eksplora.v13i2.1065.

[11] A. Alabdulatif, N. N. Thilakarathne, and M. Aashiq, "Machine Learning Enabled Novel Real-Time IoT Targeted DoS/DDoS Cyber Attack Detection System," *Computers, Materials and Continua*, vol. 80, no. 3, pp. 3655–3683, 2024, doi: 10.32604/cmc.2024.054610.

[12] F. F. Setiadi, M. W. A. Kesiman, and K. Y. E. Aryanto, "Detection of dos attacks using naive bayes method based on internet of things (iot)," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Mar. 2021. doi: 10.1088/1742-6596/1810/1/012013.

[13] K. J. Harnanta, A. Bhawiyuga, and A. Basuki, "Implementasi MQTT Broker dengan Kemampuan Auto Scaling pada Internet of Things," 2020. [Online]. Available: http://j-ptiik.ub.ac.id

[14] R. A. Khairulah, R. Herdianto, and M. A. Setiawan, "Klasifikasi Serangan Pada Jaringan Internet of Thing (IoT): Tinjauan Literatur Komparatif," *Jurnal Inovasi Teknik dan Edukasi Teknologi*, vol. 3, no. 1, pp. 47–53, doi: 10.17977/um068v3i12023p47-53.

[15] F. T. Kurdi, "Random Forest Machine Learning Technique for Automatic Vegetation Detection and Modelling in LiDAR Data," *International Journal of Environmental Sciences & Natural Resources*, vol. 28, no. 2, Jun. 2021, doi: 10.19080/ijesnr.2021.28.556234.

[16] A. Zilham Fadillah and R. Gunawan, "POTENSI IOT DALAM INDUSTRI 4.0," 2024.

[17] I. Latin and A. Transactions, “MQTT Protocol: Fundamentals, Tools and Future Directions,” 2019.

[18] R. A. Radouan Ait Mouha, “Internet of Things (IoT),” *Journal of Data Analysis and Information Processing*, vol. 09, no. 02, pp. 77–101, 2021, doi: 10.4236/jdaip.2021.92006.

[19] A. Mcewen and H. Cassimally, “Designing the Internet of Things.”

[20] E. Longo, A. E. C. Redondi, and M. Casana, *MQTT-ST: a Spanning Tree Protocol for Distributed MQTT Brokers*. IEEE, 2020.

[21] Al Hanif and R. Amri, “Implementasi Internet Of Things Pada Protokol MQTT Dan HTTP Dalam Sistem Pendeteksi Banjir,” *JURNAL INOVTEK POLBENG*, vol. 8, no. 2, 2023.

[22] R. Poorzare and A. C. Augé, “Challenges on the way of implementing TCP over 5G networks,” *IEEE Access*, vol. 8, pp. 176393–176415, 2020, doi: 10.1109/ACCESS.2020.3026540.

[23] R. Tandon, “A Survey of Distributed Denial of Service Attacks and Defenses,” Aug. 2020, [Online]. Available: http://arxiv.org/abs/2008.01345

[24] I. Wijayana and M. Ardiansyah, “Optimizing Qur’an Recitation Monitoring with Random Forest Algorithm,” *Khazanah Journal of Religion and Technology*, vol. 2, no. 1, pp. 1–5, Jun. 2024, doi: 10.15575/kjrt.v2i1.227.

[25] P. Kumala Sari and R. Randy Suryono, “KOMPARASI ALGORITMA SUPPORT VECTOR MACHINE DAN RANDOM FOREST UNTUK ANALISIS SENTIMEN METAVERSE,” 2024.

[26] R. Tri Novita, I. Gunawan, I. Marleni, O. Gregarius Grasia, and M. Nanda Valentika abcde Teknik Elektro Sekolah Tinggi Teknologi Ronggolawe Cepu Penulis Korenspondensi, “Analisis Keamanan Wifi Menggunakan Wireshark,” 2021. [Online]. Available: https://www.researchgate.net/publication/316464159_Analisis_Keama

[27] J. Lorincz, Z. Klarin, and J. Ožegović, “A comprehensive overview of tcp congestion control in 5g networks: Research challenges and future perspectives,” Jul. 01, 2021, *MDPI AG*. doi: 10.3390/s21134510.

[28] A. Primajaya and B. N. Sari, “Random Forest Algorithm for Prediction of Precipitation,” *Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)*, vol. 1, no. 1, pp. 27–31, 2018.

[29] A. Primajaya and B. N. Sari, “Random Forest Algorithm for Prediction of Precipitation,” *Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)*, vol. 1, no. 1, pp. 27–31, 2018.

[30] L. Qadrini, “Undersampling dan K-Fold Random Forest Untuk Klasifikasi Kelas Tidak Seimbang,” *Building of Informatics, Technology and Science (BITS)*, vol. 4, no. 4, Mar. 2023, doi: 10.47065/bits.v4i4.3141.

[31] M. Arief, P. Hari Trisnawan, and M. Data, “IMPLEMENTASI SISTEM DETEKSI SERANGAN SLOWLORIS PADA ARSITEKTUR JARINGAN SOFTWARE-DEFINED NETWORK MENGGUNAKAN RANDOM FOREST,” 2017. [Online]. Available: http://j-ptiik.ub.ac.id

[32] S. A. Laghari, S. Manickam, and S. Karuppayah, “A Review on SECS/GEM: A Machine-to-Machine (M2M) Communication Protocol for Industry 4.0,” *International Journal of Electrical and Electronic Engineering and Telecommunications*, vol. 10, no. 2, pp. 105–114, Mar. 2021, doi: 10.18178/ijeetc.10.2.105-114.

[33] D. Silva, L. I. Carvalho, J. Soares, and R. C. Sofia, “A performance analysis of internet of things networking protocols: Evaluating MQTT, CoAP, OPC UA,” *Applied Sciences (Switzerland)*, vol. 11, no. 11, Jun. 2021, doi: 10.3390/app11114879.

[34] H. Zi Rui, T. Ying Chien, L. Xin Ee, C. Wei San, and L. Teng Yi, “Comparison of the Use of Support Vector Machine (SVM) & Random Forest Algorithms (RF) for DDOS Attack Detection”, doi: 10.47772/IJRISS.

[35] Y. Irawan, R. Pramitasari, W. M. Ashari, A. Nur, and H. Yansyah, “Support Vector Machine Classification Algorithm for Detecting DDoS Attacks on Network Traffic,” 2025. [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAIC

[36] A. Hakeem and A. Attiah, “Machine Learning-based Approach for Detecting DDoS Attacks in Software Defined Networks,” 2024.