Defending against App-Layer DDoS: Advanced Machine Learning Security

Salim Salmi
Engineering, Systems and Applications Laboratory
Sidi Mohamed Ben Abdellah University
ENSA, Fez, Morocco
Salim.salmi@usmba.ac.ma

Lahcen Oughdir
Engineering, Systems and Applications Laboratory
Sidi Mohamed Ben Abdellah University
ENSA, Fez, Morocco
Lahcen.oughdir@usmba.ac.ma

Anas El Affar
Department of Mathematics, Physics and Informatics
Sidi Mohamed Ben Abdellah University
FP, Taza, Morocco
anas.elaffar@usmba.ac.ma

***Abstract*—Distributed denial-of-service attacks in the application layer (DDoS) pose a significant threat to online services, leading to significant financial losses and reputational damage. This intriguing research paper presents a novel automated detection approach to identify dangerous application layer DDoS attacks. Four machine learning algorithms, including random forest, decision tree, K-nearest neighbor and naive Bayes, were tested on three attack scenarios: benign (legitimate), DoS Slowloris and DoS Hulk using network traffic data feature extraction from deep packet inspection. The results showed that the Random Forest classifier excelled in all cases with incredible accuracy, precision recall, and F1 scores. The proposed approach provides a robust and efficient solution for mitigating application layer DDoS attacks, enhancing the security of online services, and reducing the impact of DDoS attacks.**

***Index Terms*—machine learning, application layer, DDoS attacks.**

I. Introduction

Distributed Denial-of-Service (DDoS) attacks are taking an increasingly serious toll on businesses, networks, and individuals across the globe. DDoS attacks involve flooding a system with an immense amount of data from multiple computers to overwhelm resources and make them unavailable to users [1]. Although all types of DDoS are difficult to detect and mitigate, application layer-focused DDoS attacks can be particularly tricky since they often mimic legitimate user behaviour. As this type of malicious attack becomes more sophisticated, it is critical that organizations take proactive steps to better protect themselves. Systems administrators should implement several layers of security and defense mechanisms, including regular patching procedures, specific user access controls, and advanced monitoring systems, to ensure that their systems are safe from these damaging cyber threats.

DDoS attacks aimed at the applications layer can have a considerable impact, seeking out weaknesses in components such as web page logins and search functionality. These malicious initiatives are often triggered by many compromised computers or botnets, making them difficult to track [2]. As a result, resource usage is stretched to its limit, leading to an overall decrease in service quality. As the threat of application layer DDoS attacks continues to rise, it is essential that efficient solutions are developed that can detect and neutralize these kinds of attacks in real time. The purpose of this paper is thus twofold: first, a machine learning analysis of application layer DDoS attacks will be conducted; second, the effectiveness of machine learning techniques in detecting such assaults will be evaluated. In addition, this research is intended to assess whether machine learning algorithms can be utilised effectively to analyse network traffic and detect application layer DDoS attacks with high degree of accuracy and low false positive rates. By exploring this possibility, it may be possible to improve current security measures.

This paper presents a new method to use machine learning algorithms to detect application layer DDoS attacks. By analyzing its performance, we can gain an understanding of existing solutions and their limitations in regard to detecting application layer DDoS attacks. Furthermore, this research will also highlight any challenges that may be encountered in this process. This paper will also analyze potential strategies to counteract application-layer DDoS attacks, with a view to offering practical guidance on their implementation.

II. Research Methodology

A. Application Layer

The seventh and top layer of the OSI (open systems interconnection) reference model, which is used to explain how computer systems communicate with each other, is the application layer. It is in charge of enabling end-to-end communication between apps and guaranteeing accurate and effective data transmission. Email, file transfers, and web services are just a few of the capabilities that the application layer offers to let programs connect with one another [14].

These services are often offered through application protocols that specify the policies and processes for data transmission between applications. Examples of such protocols are HTTP, FTP, DNS and SMTP [15]. Table I provides a description of the common application layer protocols. The application layer is also in charge of making sure that communication between apps is secure and reliable, making it a key target for various kinds of assaults, including distributed denial of service (DDoS) attacks that might prevent these services from operating normally.

TABLE I
COMMON APPLICATION LAYER PROTOCOLS [15]

Protocol	Description
HTTP	The Hypertext Transfer Protocol (HTTP) is used to transfer web content, such as HTML files and images, over the Internet.
HTTPS	HTTP Secure (HTTPS) encrypts data transferred between a web server and a web browser, providing secure web communication.
DNS	The domain name system resolves human-readable domain names (e.g., www.example.com) to IP addresses.
SMTP	The simple mail transfer protocol (SMTP) is used to send email messages between servers on the Internet.
FTP	The file transfer protocol (FTP) is used for transferring files between a client and a server on a computer network.
SSH	Secure Shell (SSH) provides secure remote access and file transfer over an unsecured network, such as the internet.

B. Classification DDoS Detection

Detecting denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks is a critical task in the field of cybersecurity. These attacks aim to disrupt the availability of a network or service by overwhelming it with traffic or exploiting vulnerabilities [16]. Classification techniques are commonly used for DDoS detection. In this section, we will explore the methods and strategies to build effective classification models for DDoS attack detection.
‘

1) Understanding DoS and DDoS Attacks: Before diving into classification techniques, it is essential to understand what DoS and DDoS attacks are:

DoS (Denial-of-Service) Attack: In a DoS attack, a single attacker or a small group of attackers attempts to make a service or network unavailable by sending an overwhelming amount of traffic or exploiting vulnerabilities. This can lead to system crashes or slowdowns [17].

DDoS (Distributed Denial-of-Service) Attack: DDoS attacks involve multiple attackers who coordinate their efforts to flood a target with traffic. These attacks are harder to mitigate because they come from various sources, making it challenging to distinguish legitimate traffic from malicious traffic.

2) A Multitude of DDoS Attack Varieties: In addition to becoming larger, DoS and DDoS attacks have also become more complex and harder to detect. Currently, these attacks can last for extended periods, and attackers often disguise their actions to look like normal internet traffic. This makes them difficult to spot using traditional methods. In this section, we will introduce different types of DoS and DDoS attacks, focusing on the ones we are studying in this research.

- **DoS Hulk** is a type of denial of service attack aimed at web servers. It involves flooding the target server with an enormous volume of HTTP requests within a very brief period. To make the attack more effective, DoS Hulk continually alters the headers and parameters in the target website's URL. This tactic bypasses caching mechanisms and compels the server to allocate as many resources as possible to handle the influx of requests, ultimately disrupting legitimate user access to the service [18].
- **DoS Slowloris** is a type of attack that operates by sending a multitude of incomplete HTTP requests to establish and maintain numerous simultaneous HTTP connections for an extended duration. This approach aims to fill up the maximum connection pool, eventually exhausting all available connections on the target server. What sets it apart from other attacks is its efficiency in terms of bandwidth usage; it can effectively disrupt services even with limited bandwidth resources. As a result, typical volumetric-based detection systems are often ineffective in identifying Slowloris attacks, primarily because the attack's volume usually does not trigger detection thresholds. One known countermeasure against this attack is implementing a timeout value to restrict the maximum duration a client can remain connected, thereby mitigating the Slowloris attack's impact [18].

C. Mathematical Model for Identifying DDoS Attacks

In this section, we present a mathematical model for the identification of (DDoS) attacks. The model leverages statistical analysis and anomaly detection techniques to distinguish between normal network traffic and malicious DDoS attacks.

Given a continuous stream of network traffic data, we aim to develop a mathematical model capable of detecting DDoS attacks in real-time. The model should provide a quantitative measure of the likelihood that incoming traffic is indicative of a DDoS attack.

1) Feature Extraction (X): We begin by extracting relevant features from incoming network traffic data. These features may include packet rates (PR), packet sizes (PS), protocol distribution (PD), and traffic volume (TV). These features collectively form the feature vector X.

2) Statistical Analysis: To establish baselines for normal network behavior, we calculate statistical measures for each feature over a defined time window. The mean (μ) and standard deviation (σ) are key statistics used to quantify these baselines:

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} x_{ij} \quad \text{for each feature } x_i \tag{1}$$

$$\sigma_i = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_{ij}-\mu_i)^2} \quad \text{for each feature } x_i \tag{2}$$

3) Anomaly Detection: Anomaly detection is performed on incoming traffic features using the calculated means and standard deviations. We can use the Z-score (Z) as a measure of deviation from the mean:

$$Z_i(x_j) = \frac{x_{ij}-\mu_i}{\sigma_i} \quad \forall \text{ feature } x_i\text{, data point } x_j \tag{3}$$

Data points with Z-scores exceeding a predefined threshold ($Z_{\text{threshold}}$) are flagged as anomalies, potentially indicative of a DDoS attack.

The output of the mathematical model is a set of anomaly scores for each incoming data point, indicating the degree to which it deviates from normal behavior. High anomaly scores suggest the presence of a DDoS attack, while low scores are indicative of normal traffic.

4) Thresholding: To classify traffic as normal or potentially a DDoS attack, we apply a threshold ($Z_{\text{threshold}}$) to the anomaly scores. Data points with anomaly scores greater than or equal to the threshold are considered suspicious and trigger alerting and mitigation mechanisms.

We have presented a mathematical model for identifying DDoS attacks based on statistical analysis and anomaly detection. The model quantifies normal network behavior, allowing it to flag anomalies that may signify the presence of a DDoS attack. By leveraging mathematical principles, this model contributes to the enhancement of network security and the timely mitigation of DDoS threats.

D. Supervised Models

In the extensive realm of machine learning models, a plethora of variations can be found in the literature, offering the potential for achieving robust classification performance. In this study, four machine learning classifiers : Random forest (RF), Decision trees (DT), Naïve Bayes (NB), and k-Nearest neighbors (KNN) were employed to classify malicious and normal traffic. Below, you will find a concise description of these utilized models for comprehensive understanding.

1) Random Forest: Random forest is an ensemble learning method that combines multiple decision trees to improve predictive accuracy and reduce overfitting [19]. The formula for a random forest prediction $\hat{y}(x)$ for a given input sample x is:

$$\hat{y}(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x) \tag{4}$$

$where$: $\hat{y}(x)$ is the predicted output for sample x and N is the number of decision trees in the forest. $T_i(x)$ is the prediction made by i.th decision tree for sample x.

Random forest combines the predictions of individual decision trees by averaging them for regression tasks or using majority voting for classification tasks. This ensemble approach enhances predictive performance and robustness [19].

2) Naive Bayes: Naive Bayes is a probabilistic classification algorithm that is based on Bayes' theorem and the "naive" assumption of feature independence. It is commonly used for text classification and spam detection [20].

Given a set of features $X = \{x_1, x_2, \ldots, x_n\}$ and a class variable C, Naive Bayes calculates the probability of a class C_k for an input vector $\mathbf{x}$ as:

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k) \cdot P(C_k)}{P(\mathbf{x})} \tag{5}$$

$where$ $P(C_k|\mathbf{x})$ is the posterior probability of class C_k given the features $\mathbf{x}$, $P(\mathbf{x}|C_k)$ is the likelihood of observing features $\mathbf{x}$ given class C_k, $P(C_k)$ is the prior probability of class C_k and $P(\mathbf{x})$ is the marginal likelihood.

The "Naive" assumption in Naive Bayes is that features are conditionally independent given the class label, simplifying the calculation of $P(\mathbf{x}|C_k)$.

Naive Bayes assigns an input vector $\mathbf{x}$ to the class C_k that maximizes the posterior probability $P(C_k|\mathbf{x})$.

3) K-Nearest Neighbors (KNN): K-Nearest Neighbor (KNN) is a supervised machine learning algorithm used for classification and regression tasks. It makes predictions based on the majority class or average value of its K nearest neighbors in the training data [21].

Given a dataset with labeled examples, let X represent the feature space, Y represent the set of class labels or target values, and D represent the training dataset.

For classification tasks, KNN predicts the class of a new input vector $\mathbf{x}$ as follows:

$$\hat{Y}(\mathbf{x}) = \text{argmax}_c \left(\sum_{i=1}^{K} [y_i = c] \right) \tag{6}$$

$where$ $\hat{Y}(\mathbf{x})$ is the predicted class for input $\mathbf{x}$, K is the number of nearest neighbors to consider and y_i is the class label of the i-th nearest neighbor.

For regression tasks, KNN predicts the value of a new input vector $\mathbf{x}$ as the average of the target values of its K nearest neighbors:

$$\hat{y}(\mathbf{x}) = \frac{1}{K} \sum_{i=1}^{K} y_i \tag{7}$$

$where$:
- $\hat{y}(\mathbf{x})$ is the predicted value for input $\mathbf{x}$. - K is the number of nearest neighbors to consider. - y_i is the target value of the i-th nearest neighbor.

KNN calculates the similarity or distance between input $\mathbf{x}$ and all data points in D and selects the K nearest neighbors to make predictions.

4) Decision Trees: Decision Trees are supervised machine learning algorithms used for both classification and regression tasks. They construct a tree-like model of decisions and their consequences [22].

Let D represent the training dataset, X represent the feature space, and Y represent the set of class labels or target values.

A decision tree recursively partitions the feature space X into regions or leaves, where each leaf corresponds to a class label (for classification) or a predicted value (for regression).

The decision tree model can be defined mathematically as a function $T(x)$ that maps an input feature vector $\mathbf{x} \in X$ to a class label y or a value:

$$T(x) = \begin{cases} y & \text{if leaf} \\ T_i(x) & \text{if node with condition } C_i \end{cases} \tag{8}$$

$where$ $T(x)$ is the prediction made by the decision tree for input feature vector $\mathbf{x}$, leaf represents a leaf node with an associated class label or value and $T_i(x)$ represents a subtree for which the condition C_i is satisfied.

A decision tree is constructed by recursively selecting the best feature to split the data at each node based on criteria such as Gini impurity or information gain. The goal is to minimize impurity or maximize information gain to improve the homogeneity of the data in each partition.

The prediction path through the tree is determined by evaluating the feature values of $\mathbf{x}$ against the conditions at each node until a leaf node is reached. Decision trees provide interpretable models and are widely used in various machine learning applications.

III. Proposed Methodology

In this section, we outline the methodology employed to develop and evaluate our automated detection approach for uncovering Application Layer Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks using machine learning techniques. The methodology is structured into the following key stages, as shown in Algorithm 1.

Algorithm 1: Proposed Classification Methodology

```
Data: Training sample data, Testing sample data
Result: Classification model and predicted labels for
        the testing sample
Step 1: Determine Labels and Extract Features for
  Training Data;
foreach sample $i$ in the training sample do
    Determine label $L_i$ by pattern analysis;
    Extract features $F_i = \{F_{i,1}, F_{i,2}, \ldots, F_{i,n}\}$;
end
Step 2: Establish the classification model;
Select an ML algorithm (RF, NB, DT, KNN);
Train the model with input features $F$ and labels $L$;
Step 3: Extract Features and Predict Labels for Testing
  Data;
foreach sample $j$ in the testing sample do
    Extract features $F_j$ in the same way as for the
      training data;
    Use the established classification model to predict
      the label $L_j$;
end
Step 4: Performance Evaluation;
Evaluate the performance of the classification model
  using metrics: accuracy, recall, precision, and the
  F1-score;
```

The available data are divided into two primary segments: a training sample designed to construct and train the classification model and a testing sample containing data awaiting classification.

Within the training sample, labels are initially assigned through pattern analysis to categorize the data into distinct groups, including DoS Hulk attacks, DoS Slowloris attacks, and benign network traffic. Subsequently, the process involves the extraction of features that encapsulate the unique characteristics of these attack types and benign traffic. We denote Fi, n as the nth feature of sample i and Li as the label assigned to sample i. This enables us to represent each sample as a feature vector, denoted as $Si = Fi, Li$, where $Fi = Fi, 1, Fi, 2, \ldots, Fi, n$.

The subsequent phase involves the establishment of a classification model using machine learning techniques with random forest (RF), naïve Bayes (NB), decision trees (DT), and k-nearest neighbors (KNN). This model is trained using the feature vectors as model inputs (F) and the expected labels as model outputs (L).

The same feature extraction process is applied to the testing sample, and the established classification model is leveraged to determine labels based on the extracted features, allowing for the detection and classification of DDoS attacks and benign traffic.

After model training, the final steps include the performance evaluation of each machine learning model. This evaluation is performed using key metrics, including accuracy, recall, precision, and the F1-score, to assess the model's effectiveness in detecting and classifying DDoS attacks and benign traffic. These metrics provide a comprehensive understanding of the model's performance and its ability to accurately distinguish between different attack types and legitimate network traffic.

A. Dataset Description

This study utilized the application layer Denial-of-Service (DDoS) dataset, sourced from Kaggle, a renowned provider of benchmark datasets [23]. The dataset comprises a total of 809,361 records, encompassing 78 attributes pertaining to application layer DDoS attacks. Through comprehensive network analysis, these records were divided into three different classes: **DoS hulk** (which denotes a DDoS attack), **Benign** (which represents legitimate traffic), and **DoS slowloris** (which indicates a DoS attack).

The distribution proportion for every single class instance is displayed in Figure 1, showing that, with 370,623, 'benign' has the greatest number of instances. Concurrently, 'DoS slowloris' has 128,612 instances, while 'DoS hulk' comprises 310,126 instances. This distribution highlights the dataset's inherent class imbalance. To address this issue, from each class, we took out 25,000 entries to create a balanced dataset.

B. Feature Selection Algorithm

In this study, we harnessed the power of four machine learning models: random forest (RF), decision tree (DT), naive Bayes (NB), and K-nearest neighbors (KNN), to steer our feature selection journey. Feature selection plays a pivotal role in pattern recognition, seeking to pinpoint the most effective subset of variables that can adeptly represent a given set of classes [24]. Our variable selection process unfolded in two strategic stages.

First, cross-validation and recursive feature elimination were used. This technique helped refine our variable set, ensuring that only the most pertinent features made the cut.

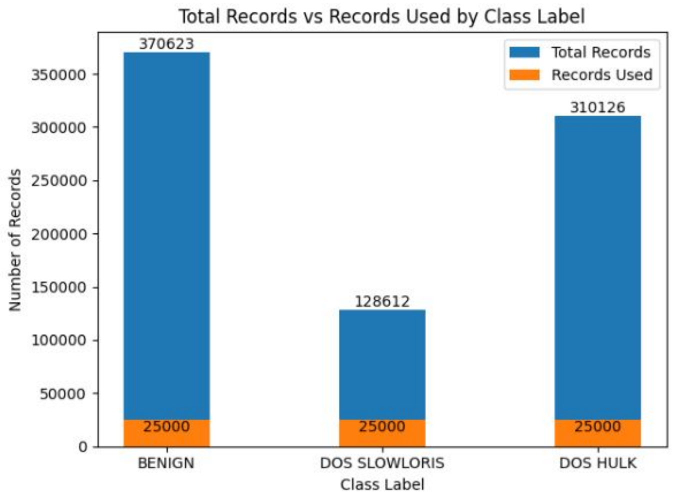


Fig. 1. Total and used records

In the second stage, we embarked on a novel feature selection journey using the random forest approach. This innovative method proved to be efficient, resulting in a reduction of the initial 78 variables to a lean and potent 20. This trimmed set of variables even brought about a slight uptick in accuracy, further validating our approach.

We conducted this process with specific input parameters in mind: 2500 rounds, a variable importance threshold of 99%, a global accuracy threshold of 98%, and a per-class accuracy requirement of 89%. These parameters helped us streamline our selection, ensuring that we maintained a high level of accuracy. It is interesting to note that while most models adopted the 20 selected variables, each model had a unique set of preferred variables. To sift through these choices and pinpoint the most impactful variables from our selected models, we relied on the RF variable importance criterion, as elaborated in Algorithm 2. The final outcome of our feature selection process is visually depicted in Figure 2.

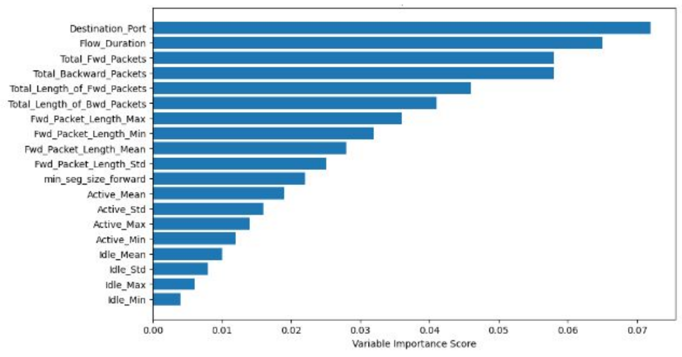


Fig. 2. Variables selected using our proposed feature selection algorithm

In the final steps of the feature selection process, the models are organized by the number of variables they utilize. Outliers are subsequently eliminated from this grouped model set. Then, the group of models characterized by the highest occurrence frequency, together with the associated variable count labeled "N", is selected. Following this selection, the variables are ranked based on their mean importance, as calculated in a previous step. Finally, the feature selection process culminates by returning the "N" most important variables as the ultimate selection, ensuring a streamlined and optimized set of features for the machine learning model.

Algorithm 2: Feature Selection Algorithm

```
Input:
  • Database Characteristics
  • Level of importance
  • Precision threshold
  • Number of rounds
Output: Selected variables
for rounds ← 1 to Number of rounds do
    CurrentVariables ← All descriptors
    while True do
        Split the dataset for testing and training. ;
        Train TrainModel using CurrentVariables ;
        Identify the most important variables from TrainModel ;
        Calculate cumulative importance ;
        if Cumulative importance < Importance threshold then
            Exit loop ;
        end
        Train TrainModel with most essential features ;
        Test TrainModel and calculate accuracy ;
        if Accuracy < Accuracy threshold then
            Exit loop ;
        end
        Record TrainModel as OptimizedModels ;
        CurrentVariables ← Most important variables ;
    end
end
```

C. Model Hyperparameters

Model Hyperparameters are predefined configuration settings and parameters that control various aspects of the training and behavior of a machine learning model. Unlike model parameters, which are learned from the training data, hyperparameters are set before training and guide the learning process. These hyperparameters influence the model's architecture, optimization process, and overall performance, and they are typically tuned through experimentation to achieve the best model performance on a specific task [25], Table II provides a summary of the hyperparameters for machine learning models.

D. Performance Metrics

Evaluating machine learning models for network attack detection [26] using key metrics:

- **Accuracy**: Overall correctness. - **Precision**: Accuracy of positive predictions, minimizing false positives. - **Recall**: True

TABLE II
HYPERPARAMETERS OF MACHINE LEARNING MODELS

Model	Hyperparameters
NB	estimator=GaussianNB(), verbose=1, cv=10, n_jobs=None
DT	criterion=entropy, max_depth=3, NOC=8
RF	criterion: entropy, max_depth: 5, n_estimators: 100
KNN	metric: euclidean, n_neighbors: 150, weights: distance

positive rate, minimizing false negatives. - **F1-Score**: Balance between precision and recall, suitable for imbalanced datasets.

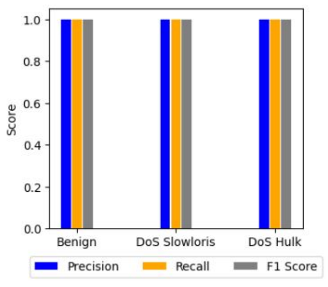
These metrics assess the model's ability to distinguish benign and DDoS attack traffic at the Application layer.

IV. RESULTS AND DISCUSSION

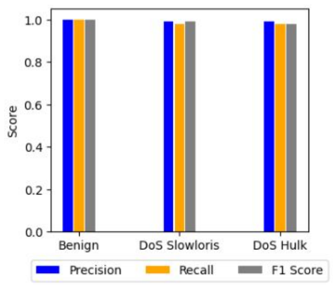
In this section, we delve into the outcomes of our experiments focused on the detection of DDoS attacks. To accommodate this objective, we conducted a series of experiments utilizing various machine learning models. These experiments were conducted on a dataset encompassing an application layer DDoS attacks.

A. *performance using all features*

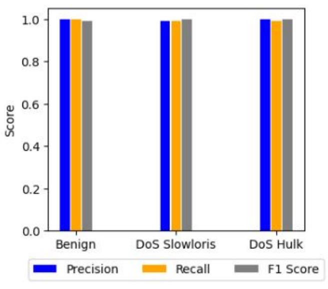
To differentiate between legitimate and malicious interactions, machine learning algorithms were used in the first phase of examinations with no sort of feature selection techniques. A comprehensive set of 78 attributes was utilized for these experiments, The accompanying histogram in Figure 3 presents a comprehensive overview of the performance metrics for four machine learning models : random forest, decision tree, naive Bayes, and K-nearest neighbors. The models were assessed based on four essential metrics: accuracy, precision, recall, and F1-score.



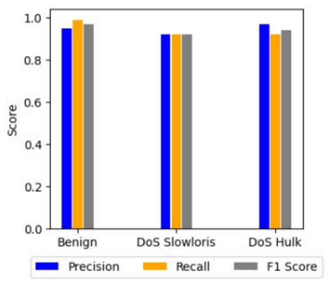
(a) RF



(b) DT



(c) NB



(d) KNN

Fig. 3. Comparing Performance Using All Features

In this evaluation, random forest emerges as the undeniable champion, achieving a perfect score of 1.0 on all metrics. This remarkable feat underscores its exceptional ability to deliver consistently accurate predictions with impeccable precision, while its recall and F1-score metrics also demonstrate its prowess in correctly identifying positive instances while maintaining a harmonious balance between precision and recall. The decision tree model secures the second position with commendable performance, attaining an accuracy of 0.99, signifying its proficiency in classifying data accurately, and showcasing its capability to minimize false positives with a precision score of 0.99. While naive Bayes exhibits reliability and a noteworthy capacity for accurate predictions with an accuracy of 0.99, it falls slightly short of the top two models in terms of precision and recall balance. K-nearest neighbors (KNN) rounds out the evaluation, demonstrating an accuracy of 0.92 and strength in minimizing false positives with a precision score of 0.92, albeit with room for improvement in achieving a more balanced trade-off between precision and recall.

Table III provides a performance comparison based on 25k training samples for each class of attacks. Among the classifiers, random forest (RF) consistently demonstrates the greatest F1 score, recall, and precision across all classes. Conversely, decision trees (DT) and naïve Bayes (NB) exhibit slightly lower performance. Notably, k-nearest neighbors (KNN) exhibits the weakest performance, particularly when dealing with DDoS attacks.

TABLE III
EVALUATING PERFORMANCE USING ALL FEATURES

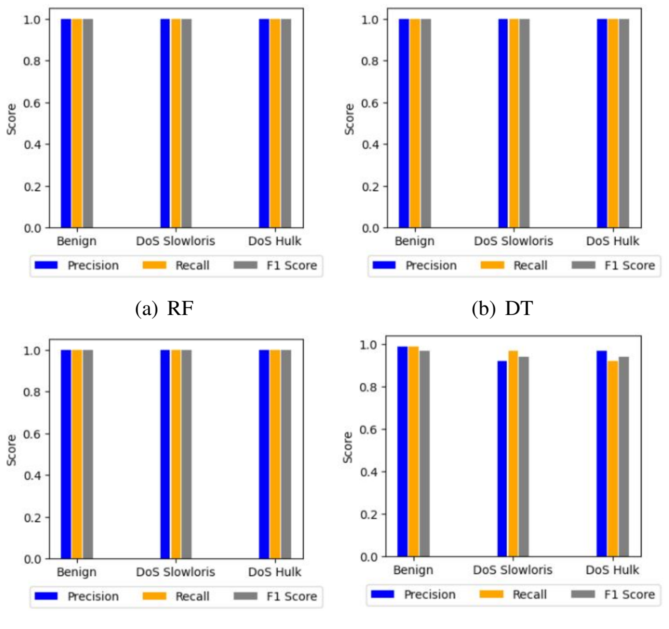
Classifier	Class	Precision	Recall	F1 Score
RF	Benign	1.00	1.00	1.00
	DoS Slowloris	1.00	1.00	1.00
	DoS Hulk	1.00	1.00	1.00
KNN	Benign	0.95	0.99	0.97
	DoS Slowloris	0.92	0.92	0.92
	DoS Hulk	0.97	0.92	0.94
DT	Benign	1.00	1.00	1.00
	DoS Slowloris	0.99	0.98	0.99
	DoS Hulk	0.99	0.98	0.98
NB	Benign	1.00	1.00	0.99
	DoS Slowloris	0.99	0.99	1.00
	DoS Hulk	1.00	0.99	1.00

B. *Performance using Multi-Features*

To find and pick key features in the data samples, we used feature selection approaches in the next set of tests. The integration of these feature selection techniques led to notable enhancements in the performance of our machine learning models. As a result, we observed improvements across all learning models, which are documented in Table IV. In particular, the utilization of multiple features significantly elevated the accuracy scores of our RF, NB and DT models. Furthermore, the performance of the K-nearest neighbors

(KNN) classifier also experienced enhancements through these techniques.

Figure 4 illustrates that all machine learning models attain their highest levels of performance when trained using multi-features extraction . Notably, among the various models examined, random forest (RF) consistently outperforms others across all categories of network attacks.



(a) RF (b) DT (c) NB (d) KNN

Fig. 4. Comparing Performance Using Multi-Features

Table IV underscores the significance of feature engineering techniques in enhancing the effectiveness of machine learning models and optimizing results. Our experimental findings underscore that ensemble models, exhibit the highest accuracy due to their utilization of multiple decision trees, often outperforming single decision tree-based models in the classification process. It is worth noting that our study encompasses different target classes, comprising 'benign', 'DoS Slowloris' and 'DoS Hulk' and the accuracy varies across individual classes. Specifically, K-nearest neighbors (KNN) records the lowest accuracy of 0.92 for DoS Slowloris attack detection, marking the lowest accuracy among all the models.

C. K-Fold Cross Validation

To underscore the importance of the suggested multi-features method, we conducted a 10-fold cross validation [27]. The results of this 10-fold cross validation, performed with 25,000 samples, are presented in Table V. The outcomes reveal that the random forest (RF), decision trees (DT), and naïve Bayes (NB) models achieved remarkable accuracy with a standard deviation of (+/- 0.00) in the cross-validation process. Conversely, the K-nearest neighbors (KNN) model obtained an accuracy score of 0.96.

TABLE IV
EVALUATING PERFORMANCE USING MULTI-FEATURES

Classifier	Class	Precision	Recall	F1 Score
RF	Benign	1.00	1.00	1.00
	DoS Slowloris	1.00	1.00	1.00
	DoS Hulk	1.00	1.00	1.00
KNN	Benign	0.99	0.99	0.99
	DoS Slowloris	0.92	0.97	0.94
	DoS Hulk	0.97	0.94	0.94
DT	Benign	1.00	1.00	1.00
	DoS Slowloris	1.00	1.00	1.00
	DoS Hulk	1.00	1.00	1.00
NB	Benign	1.00	1.00	1.00
	DoS Slowloris	1.00	1.00	1.00
	DoS Hulk	1.00	1.00	1.00

TABLE V
ACCURACY OF K-FOLD CROSS VALIDATION

Model	No. of features	Accuracy	Standard Deviation
RF	20	1.00	(±0.00)
KNN	20	0.96	(±0.00)
DT	20	1.00	(±0.00)
NB	20	1.00	(±0.00)

D. Contrasting with Previous Research

In the context of comparing our approach with existing studies, we conducted a performance evaluation to highlight the effectiveness of our multi-features approach for detecting DDoS attacks. To facilitate this comparison, we implemented and tested models from recent, state-of-the-art studies using the same dataset employed in our research. Specifically, we selected the most up-to-date studies addressing similar tasks for this evaluation.

We rigorously followed the implementation details provided by these studies and executed experiments under the same experimental conditions used in our proposed approach. The outcomes of this performance comparison are summarized in Table VI.

TABLE VI
STUDYING COMPARISONS WITH RELATED WORKS

Reference	Year	Model	Accuracy
[3]	2017	SVM	99.4
[4]	2018	DFF	99.63
[5]	2019	RNN;SVM	99.74;99.99
[6]	2022	Looking-Back RF	99.81
[7]	2022	Tree-Layer	98.8
[8]	2022	XGBoost+ANOVA	98.35
[9]	2019	RF;NB;J48	99.4;99.3;99.33
[10]	2022	LSTM	98.88
[11]	2020	RNN+AE	99.0
[12]	2020	SVM;MLP;DT;RF	92.4;92.5;99.0;99.0
[13]	2019	CNN+FNN	77.84
Our	2023	**RF;DT;NB;KNN**	**1.00;1.00;1.00;0.94**

Our findings indicate that our proposed approach significantly outperforms existing methods. This superiority is attributed to its streamlined architecture and consistent, robust results. By incorporating cutting-edge techniques along with feature selection, we achieved both high accuracy and efficiency. Notably, the majority of the models achieved 100% accuracy scores when leveraging our multi-feature approach.

V. Conclusion

In conclusion, our automated approach effectively detects application layer DDoS attacks, showcasing strong performance across diverse scenarios. The random forest classifier consistently achieves high accuracy, precision, recall, and F1 scores, underscoring its effectiveness. Future work includes exploring advanced feature extraction, expanding datasets, and focusing on real-time adaptive detection mechanisms to counter evolving attack strategies.

References

[1] Bonguet A, Bellaiche M. A Survey of Denial-of-Service and Distributed Denial of Service Attacks and Defenses in Cloud Computing. Future Internet. 2017; 9(3):43. https://doi.org/10.3390/fi9030043.

[2] Adedeji, K.B.; Abu-Mahfouz, A.M.; Kurien, A.M. DDoS Attack and Detection Methods in Internet-Enabled Networks: Concept, Research Perspectives, and Challenges. J. Sens. Actuator Netw. 2023, 12, 51. https://doi.org/10.3390/jsan12040051

[3] Alshamrani, A.; Chowdhary, A.; Pisharody, S.; Lu, D.; Huang, D. A defense system for defeating DDoS attacks in SDN based networks. In Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access, Miami, FL, USA, 21–25 November 2017; pp. 83–92.

[4] Khuphiran, P.; Leelaprute, P.; Uthayopas, P.; Ichikawa, K.; Watanakeesuntorn, W. Performance comparison of machine learning models for DDoS attacks detection. In Proceedings of the 22nd IEEE International Computer Science and Engineering Conference, Chiang Mai, Thailand, 21–24 November 2018; pp. 1–4.

[5] Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. Future Gener. Comput. Syst. 2019, 100, 779–796.

[6] Mihoub, A.; Fredj, O.B.; Cheikhrouhou, O.; Derhab, A.; Krichen, M. Denial of service attack detection and mitigation for internet of things using looking-back-enabled machine learning techniques. Comput. Electr. Eng. 2022, 98, 107716

[7] Aslam, M.; Ye, D.; Tariq, A.; Asad, M.; Hanif, M.; Ndzi, D.; Chelloug, S.A.; Elaziz, M.A.; Al-Qaness, M.A.; Jilani, S.F. Adaptive machine learning based distributed denial-of-services attacks detection and mitigation system for SDN-enabled IoT. Sensors 2022, 22, 2697

[8] Gaur, V.; Kumar, R. Analysis of machine learning classifiers for early detection of DDoS attacks on IoT devices. Arab. J. Sci. Eng. 2022, 47, 1353–1374

[9] Sharma, M.; Elmiligi, H.; Gebali, F.; Verma, A. Simulating attacks for RPL and generating multi-class dataset for supervised machine learning. In Proceedings of the IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference, Vancouver, BC, Canada, 17–19 October 2019; pp. 20–26.

[10] Priyadarshini, R.; Barik, R.K. A deep learning based intelligent framework to mitigate DDoS attack in fog environment. J. King Saud Univ. Comput. Inf. Sci. 2022, 34, 825–831.

[11] Elsayed, M.S.; Le-Khac, N.A.; Dev, S.; Jurcut, A.D. Ddosnet: A deep-learning model for detecting network attacks. In Proceedings of the IEEE 21st International Symposium on A World of Wireless, Mobile and Multimedia Networks, Cork, Ireland, 31 August–3 September 2020; pp. 391–396.

[12] Santos, R.; Souza, D.; Santo, W.; Ribeiro, A.; Moreno, E. Machine learning algorithms to detect DDoS attacks in SDN. Concurr. Comput. Pract. Exp. 2020, 32, e5402.

[13] Zhu, M.; Ye, K.; Xu, C.Z. Network anomaly detection and identification based on deep learning methods. In Cloud Computing—CLOUD 2018; Lecture Notes in Computer Science; Luo, M., Zhang, L.J., Eds.; Springer: Cham, Switzerland, 2018.

[14] Conrad, E., Misenar, S., & Feldman, J. (2016). Chapter 5—Domain 4: Communication and network security (designing and protecting network security). CISSP Study Guide, 219-291.

[15] Wenhua, Z., et al.: Data security in smart devices: advancement, constraints and future recommendations. IET Netw. 1–13 (2023). https://doi.org/10.1049/ntw2.12091.

[16] Hamarshe, A., Ashqar, H.I., Hamarsheh, M. (2023). Detection of DDoS Attacks in Software Defined Networking Using Machine Learning Models. In: Daimi, K., Al Sadoon, A. (eds) Proceedings of the 2023 International Conference on Advances in Computing Research (ACR'23). ACR 2023. Lecture Notes in Networks and Systems, vol 700. Springer, Cham. https://doi.org/10.1007/978-3-031-33743-7_51.

[17] Al-Abadi, A. A. J., Mohamed, M. B., & Fakhfakh, A. (2023) . Robust and Reliable Security Approach for IoMT: Detection of DoS and Delay Attacks through a High-Accuracy Machine Learning Model.

[18] M. Goel, S. Singh, A. Garg and N. R. Roy, "Comparative Study of DDoS Attacks & Tools and Their Analysis," 2023 International Conference on IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2023, pp. 1-8, doi: 10.1109/ICICAT57735.2023.10263744.

[19] Speiser, J. L., Miller, M. E., Tooze, J., & Ip, E. (2019). A comparison of random forest variable selection methods for classification prediction modeling. Expert systems with applications, 134, 93-101.

[20] Chen, S., Webb, G. I., Liu, L., & Ma, X. (2020). A novel selective naïve Bayes algorithm. Knowledge-Based Systems, 192, 105361.

[21] K. Taunk, S. De, S. Verma and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), Madurai, India, 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.

[22] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning", JASTT, vol. 2, no. 01, pp. 20 - 28, Mar. 2021.

[23] Application Layer DoS Attack Dataset. Available online: https://www.kaggle.com/hamzasamiullah/ml-analysis-application-layer-dos-attack dataset, (accessed on 30 August 2021).

[24] Khaire, U. M., & Dhanalakshmi, R. (2022). Stability of feature selection algorithm: A review. Journal of King Saud University-Computer and Information Sciences, 34(4), 1060-1073.

[25] Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, 295-316.

[26] Handelman, Guy S., et al. "Peering into the black box of artificial intelligence: evaluation metrics of machine learning methods." American Journal of Roentgenology 212.1 (2019): 38-43.

[27] Wong, T. T., & Yeh, P. Y. (2019). Reliable accuracy estimates from k-fold cross validation. IEEE Transactions on Knowledge and Data Engineering, 32(8), 1586-1594.