International Journal of Computer Applications (0975 - 8887)
Volume 186 - No.43, September 2024

Machine Learning-based Approach for Detecting DDoS Attacks in Software Defined Networks

Abeer Hakeem
Department of Information Technology,
Faculty of Computing and Information Technology
King Abdulaziz University, Jeddah, Saudi Arabia

Afraa Attiah
Department of Information Technology,
Faculty of Computing and Information Technology
King Abdulaziz University, Jeddah, Saudi Arabia

ABSTRACT

Software-Defined Networking (SDN) provides enhanced manageability, control, and dynamic updating of network rules through the separation of the control and data planes. However, SDN architectures remain vulnerable to various network attacks, including Distributed Denial of Service (DDoS) attacks. To address this challenge, this paper proposes the DDoSDetect solution, which leverages Logistic Regression machine learning algorithm to detect DDoS attacks in SDN environments. The DDoSDetect solution focuses on identifying flooding-based DDoS attacks, including TCP SYN, HTTP, UDP, and ICMP attacks, by analyzing SDN network traffic. The Logistic Regression classifier is trained to distinguish between normal and attack traffic based on four key features: number of packets, packet size, source and destination MAC addresses. The performance of the DDoSDetect solution is evaluated and compared to other binary classification algorithms, such as Naive Bayes, Random Forest, K-Nearest Neighbor and Support Vector Machine. The experimental results demonstrate that the DDoSDetect solution based on logistic regression outperforms the well-known performing alternative classifiers, achieving an accuracy improvement of 2.4%, an F1-score enhancement of 2.0%, and a precision increase of 11.68%.

Keywords

DDoS, SDN, Logistic Regression, Naive Bayes, Random Forest, and Support Vector Machine Machine learning, and Feature Selection.

1. INTRODUCTION

The rapid development of new technologies has highlighted the need to reconfigure network policies and upgrade devices. Conventional networks have static architectures that cannot meet these requirements. In traditional networks, the control plane, forwarding plane and application plane are tightly coupled, making dynamic network management difficult.

Introducing new features or upgrading existing ones require manual reconfiguration of all network devices, a tedious and costly process [1]. Additionally, these devices are vendor-specific, requiring separate configuration, and the software, or network operating system, accounts for a significant portion of the high device costs.

Software-Defined Networking (SDN) solves the problems of traditional networks by simplifying and centralizing network management [2, 3]. The key advantages of SDN are programmability, vendor neutrality, greater agility, centralized supervision, and network automation. This allows administrators to manage multiple devices from a centralized controller rather than configuring each one individually. It also enables easy separation of experimental and production traffic without interference. SDN is more cost-effective than traditional networks, as a single controller can manage all devices, reducing deployment and management expenses.

According to a Markets report, the SDN market is expected to grow from $8.8 billion in 2018 to $28.9 billion by 2023, a 26.8% [4]. The primary growth driver is the need for advanced network management to handle increasing traffic and complexity. However, SDN faces concerns around network security, scalability, and supportability. Of these, security is a primary issue. The centralized SDN controller is responsible for managing the network, so its failure would disrupt the entire system. Additionally, the centralized control and communication between the controller and switches could be targets for sophisticated Distributed Denial of Service (DDoS) attacks. The goal of DDoS attacks is to consume the resources of SDN components, leading to network performance degradation.

The distributed and variable nature of DDoS attacks, including their shifting volumes and use of spoofed IP addresses, make them difficult to detect and address [5]. DDoS threats target all layers of the SDN architecture. At the data plane, the OpenFlow-enabled forwarding devices (known as OpenFlow switches) have limited flow table sizes to store rules and limited processing capacity to handle unmatched packets [6]. This makes the data plane susceptible to DDoS attacks like buffer saturation and flow table overloading. Further, adversaries can overload the control plane by saturating its computational and network resources [7]. The application plane is also vulnerable, as DDoS attacks can target the northbound API or launch traditional application-layer DDoS assaults [8]. Addressing these multi-layered DDoS threats in SDN environments requires a comprehensive security strategy.

A wide range of techniques have been used in conventional networks to reduce the effect of DDoS attacks, but these often proved resource-intensive [9]. For example, packet analysis in traditional

networks required significant processing power, so sampling techniques were used to verify packets instead, like Cisco's NetFlow and SFlow technologies for traffic collection and analytics [10]. The programmable nature of software-defined networking (SDN) offers opportunities for more dynamic DDoS mitigation. When attacks are detected, flow rules can be dynamically inserted into the SDN switch flow tables to address the malicious traffic [11]. Many DDoS defense mechanisms in SDN leverage statistical analysis and machine learning techniques, taking advantage of the flow statistics provided by the OpenFlow southbound API. This enables communication between switches and the SDN controller allows extracting relevant features for security analytics to detect DDoS attacks [12].

The main goal of this paper is to detect DDoS attack by leveraging logistic regression algorithm in SDN environment. Logistic regression is particularly well-suited for binary classification tasks, making it a natural choice for distinguishing between normal and attack traffic [13]. The interpretability of logistic regression model can provide insights into the relative importance of different features in the DDoS detection process. The solution focuses on identifying flooding-based DDoS attacks, including TCP SYN, HTTP, UDP, and ICMP attacks, by analyzing SDN network traffic. The logistic regression classifier is trained to distinguish between normal and attack traffic based on four key features: number of packets, packet size, source MAC address, and destination MAC address. The performance of the DDoSDetect solution is evaluated and compared to other binary classification algorithms, such as Naive Bayes, Random Forest, K-Nearest Neighbor, and Support Vector Machine. The experimental results demonstrate that the DDoSDetect solution based on logistic regression outperforms the well-known alternative classifiers. This indicates that the logistic regression-based DDoSDetect solution is an effective approach for detecting DDoS attacks in SDN environments.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents a design of the DDoSDetect Solution. A model is presented in Section 4. The implementation and evaluation are illustrated in Section 5. The paper concludes in Section 6.

2. RELATED WORK

Several recent studies have proposed novel frameworks and techniques for detecting DDoS attacksin software-defined networks (SDNs) using machine learning approaches.

Kang et al. [14] have proposed a way to describe DDoS attacks in terms of similarity and hierarchy. The results show that the classification is effective for dealing with attack samples based on different mechanisms and supports more than 12 attack samples. In addition, it showed that high accuracy of the results was achieved, but large datasets are not acceptable due to time complexity. Barki et al. [15] used K-Nearest Neighbors KNN cluster algorithm to classify traffic as normal and malicious. The results showed that KNN had the highest detection rate of 90% and had a good level of accuracy. However, the training took a long time. Dong et al. [16] used KNN, but the results show that the KNN algorithm have an accuracy of 85%. Yadav et al. [17] have used logistic regression to find an efficient solution to discover the traffic in DDoS application-layer attacks. The proposed method effectively classifies the attack traffic from the normal traffic with an average detection rate of 98.64% and an average false positive rate of 1.41%. Meti et al. [18] have used different machine learning algorithms to build a prediction model in SDN. When the server receives a new request from the customer, the order is sent to the model and forecasts if the connection is normal or malicious. They used Naïve Bayes NB, Support Vector Machine SVM, and Neural Network machine learning algorithms to detect DDoS attacks in Software Defined Networking. Naïve Bayes had an accuracy of 70%, while SVM and Neural Network had an 80% accuracy. Thus, the logistic regression outperformed NB and SVM in distinguishing malicious traffic from normal traffic.

Authors in [19] used an SDN dataset with normal and malicious flows. Five features are extracted from the traffic trace average (number of packets per flow, number of bytes per flow, timeout per flow-rule, percentage of pair flows, and rule FlowMod disparity). The samples are classified using six machine learning algorithms such as Logistic Regression LR, K-Nearest Neighbors KNN, Naive Bayes NB, Decision Tree DT, Random Forest RF, and Support Vector Machine SVM to classify different DDoS attacks such as Smurf, UDP flood, and HTTP flood. The average accuracy achieved by LR is 98.65% . On the other hand, RF achieved 98.40% with less execution time than LR. Ye et al. [20] developed an SDN simulation platform using Mininet, where they extracted 6-tuple characteristic values from the switch flow table (e.g., source IP speed, source port speed, flow packet standard deviation, flow byte deviation, flow entry speed, and pair-flow ratio). They then built a DDoS attack model using the SVM classification algorithm, which achieved a mean accuracy of 95.24% in distinguishing normal and malicious traffic. The proposed DDoSDetect solution apply the Logistic Regression algorithm with only four features while achieve a mean accuracy of 97%.

Bawany et al. [21] evaluated the performance of various machine learning algorithms, including KNN, SVM, NB, DT, RF, for DDoS attack detection. Their results showed that DT and RF achieved the highest accuracy, both reaching 99%. Nadeem et al. [22] created an SDN environment and extracted features such as duration, source bytes, and destination bytes to detect DDoS attacks. Using the RF algorithm, they reported an accuracy of 99.97%. The reviewed studies highlight the effectiveness of supervised machine learning techniques, such as DT and RF, in achieving high accuracy for DDoS attack detection in SDNs. In contrast, unsupervised learning approaches may face challenges in accurately sorting data and handling large datasets. Supervised detection algorithms allow for data collection, output generation, and performance optimization to address various real-world computational problems. However, the study did not consider the logistic Regression algorithm in the comparison although it is a supervised detection algorithm.
Bhayo et al. [23] introduced a framework that integrates machine learning and software-defined IoT for effective and timely detection of DDoS attacks. Operating on the SDN-WISE controller, the ML-based module employs supervised learning classifiers such as NB, SVMs, and DT to identify malicious traffic flows. Their framework achieved high accuracy rates of 98.1%, 97.4%, and 96.1% for the respective classifiers. Through a comprehensive analysis of parameters like IoT nodes, attack nodes, payload size, and packet frequency, the framework evaluates network performance based on CPU usage, attack detection time, and memory consumption. In contrast, Alubaidan et al. [24] introduced an intrusion detection technique using machine learning in the SDN environment, testing various supervised classifiers like SVM, LR, KNN, RF, and LSTM. Their study incorporated the LR classifier, which achieved an various accuracy rate for different classifiers in identifying DDoS attacks.

Elubeyd et al. [25] proposed a hybrid deep learning algorithm for detecting and defending against DoS/DDoS attacks in SDNs by combining both supervised and unsupervised techniques. Their ap-

proach achieved accuracy rates of 99.81% and 99.88% on different datasets, considering both volumetric and low-rate DDoS attacks to maintain network integrity and availability. Using hybrid methods over supervised approaches like logistic regression can present challenges in data analysis. While hybrid models offer advanced capabilities in detecting complex patterns, they often require more complex architectures, increased computational resources, and can be harder to interpret due to their black-box nature.

3. DESIGN OF THE DDOSDETECT SOLUTION

This section describes the key components of the Software Defined Network SDN and the planes in the DDoSDetect model, which utilizes Logistic Regression for identifying Distributed Denial-of-Service (DDoS) attacks in network traffic.

—**Part-A: OpenFlow Controller (RYU Controller):** The RYU controller serves as the OpenFlow controller, which manages the network topology and enforces policies.

—**Part-B: Network Topology:** This component represents the physical network infrastructure, including switches, routers, and hosts, along with the data transmission processes between them.

—**Part-C: Applications:** This layer includes various applications that leverage the SDN capabilities, such as intrusion prevention/detection systems, security systems, traffic monitoring, load balancing, bandwidth management, and quality of service.

The communication between the RYU controller (Part-A) and the network switches is facilitated by the OpenFlow protocol, which is a programmable network protocol for the SDN environment. An OpenFlow switch is an OpenFlow-enabled data switch that communicates with the external controller over the OpenFlow channel. The OpenFlow switches perform packet lookup and forwarding according to their flow tables and group tables, which are set up and managed by the RYU controller.

As shown in Figure 1, the DDoSDetect solution consists of (1) Data plan which includes the hosts that are linked to OpenFlow switches; (2) Control plane which is part of the SDN controller where packets are received and classified into the traffic into normal and malicious; (3) Application layer in the SDN architecture includes a DDoS Attack Detection application that implements the Logistic Regression method, programmed in the Python programming language. The key components of the application are as follows:

—Feature Engineering: The application employs a comprehensive feature engineering process to extract relevant network-level characteristics that can effectively capture the patterns and indicators of DDoS attacks. This includes exploring a range of metrics, such as packet rates, packet size, and other relevant network parameters.

—Logistic Regression Modeling: The core of the DDoS attack detection application is the Logistic Regression model, which learns the relationship between the extracted network features and the presence of a DDoS attack. This allows the model to accurately classify the incoming network traffic as either normal or attack traffic.

—Model Evaluation and Benchmarking: The DDoS attack detection application undergoes a thorough evaluation process using simulated network datasets. The performance of the Logistic Regression-based approach is assessed in terms of detection accuracy, false positive rates, and other relevant metrics. Additionally, a comparative analysis is conducted with alternative machine learning techniques to highlight the advantages and practical applicability of the proposed DDoS attack detection solution.

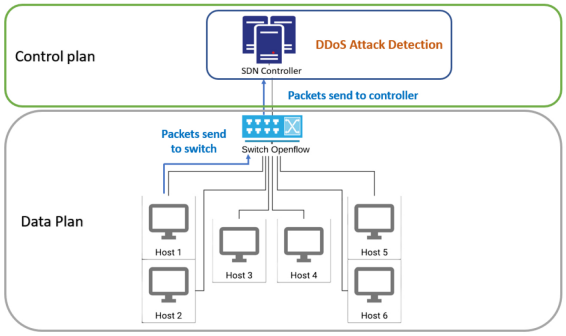


Fig. 1. The Architecture of DDoSDetect Solution

4. LOGISTIC REGRESSION MODEL

This section outlines the four major stages of the proposed DDoS-Detect solution based on Logistic Regression:

4.1 Data Collection

The first step involves collecting a comprehensive dataset of network traffic, including both normal traffic and DDoS attack traffic. Thus, a network traffic dataset is generated. The dataset built by gathering information about various network traffic parameters that can be used to analyze and detect DDoS attacks. These parameters include:

—Number of packets received by each switch

—Packet size

—Source MAC address

—Destination MAC address

The data acquisition process is carried out using a Python script that runs the RYU controller and monitors the network traffic for 30 minutes. This allowed us to capture both suspicious and normal traffic information between the hosts in the SDN.

To generate the DDoS traffic, hping3 tool is utilized to send DDoS traffic between two hosts and collect the relevant traffic information. Each normal traffic flow from the source to the destination was captured and labeled as normal traffic. In total, the collected dataset comprises 11,232 data records, with 3,412 records representing malicious DDoS traffic and 7,820 records representing normal traffic.

The generated network traffic dataset for the DDoSDetect solution is summarized as fellows:
Datapath: identifies each connection between the OpenFlow switch and the controller (i.e., Four connections).
In-port: indicates the input port through which the network traffic enters the switch.
Out-port: specifies the output port through which the network traffic leaves the switch.
Packets No.: shows the number of packets in the network traffic flow.
Bytes: represents the total number of bytes in the network traffic flow (i.e., packets size).
Traffic: classifies the network traffic as either ”normal” or ”malicious” (DDoS attack).

4.2 Data Prepossessing and Feature Selection

Data preprocessing is essential to convert the dataset into usable information for training the proposed DDoSDetect model. It aims to efficiently detect the dependent variable (traffic type) by utilizing independent variables effectively. In this phase, Jupyter Lab [26] is used to prepare the dataset to build and train the Logistic Regression model, as seen in the following steps:

First: the collected dataset is imported as a CSV file to extract the dependent and independent variables.
Second: split the dataset into source and targets.
Third: three different scenarios are implemented to split the dataset:

(1) 80% of the data is used for training, and the remaining 20% is used for testing.
(2) 70% of the data is used for training, and the remaining 30% is used for testing.
(3) 60% of the data is used for training, and the remaining 40% is used for testing.

Fourth: apply standardization feature scaling when building the Logistic Regression model to get accurate prediction results.
Fifth: train the logistic regression model for all different scenarios.
Sixth: the trained model in the previous steps is saved as a file using the Pickle library, allowing it to be used in the detection process.

4.3 Feature Extraction

The logistic regression classification algorithm is adopted to extract traffic features in order to classify traffic categories as normal or malicious traffic. Therefore, data preprocessing techniques are utilized, such as scaling and pipeline techniques, to build an accurate logistic regression model.

Standard Scaler Techniques: a standardization feature scaling is used to get accurate prediction results. The feature scaling technique used is called standardization, which is the process of changing the scale of certain features to a common one such that they have a mean value of 0 and a standard deviation of 1. It assumes that all features have the same variance and are centered around zero. If a feature has a high variance, it can dominate the objective function and prevent the estimator from learning from other features properly. Standardization is useful with classification techniques in logistic regression [27]. The standard score of a sample X is calculated as defined in Equation. 1, where U represents the mean and S represents the standard deviation of the training samples.

$$Z = \frac{X - U}{S} \tag{1}$$

Pipeline: the pipeline method is used to apply Standard Scaler and to implement fit and transform scale methods on a trained dataset to train a logistic regression model. It assembles several steps that are cross-validated together while utilizing different parameters via their names. The purpose of the pipeline is to avoid data leakage through data preparation.

4.4 Classification Processes

A Logistic Regression model has been trained to analyze and predict the category of network traffic as normal or malicious. Logistic Regression is suitable for binary classification by applying the logistic (sigmoid) function, which takes feature values as input and calculates an output between 0 and 1. If the sigmoid output is greater than 0.5, the traffic is classified as normal; if it's less than 0.5, the traffic is classified as malicious.

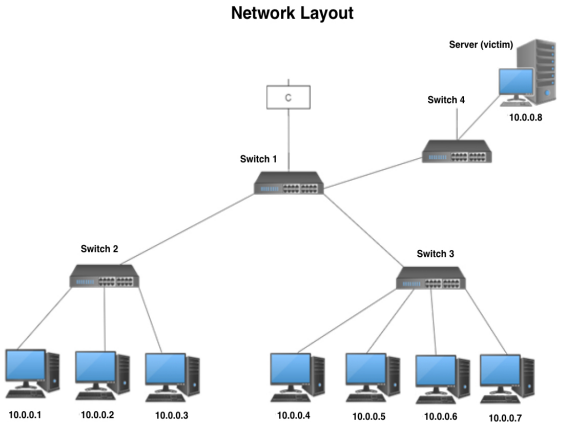


Fig. 2. Network Topology and Communication with Controller.

The logistic regression formula is based on linear regression, defined in Eq. 2, where m is the intercept and c is the slope of the line.

$$Y = mx + c \tag{2}$$

To convert the linear regression output, which can range from negative to positive infinity, to a probability between 0 and 1, the logarithm of the equation is used to define the sigmoid function, as shown in Eq. 3:

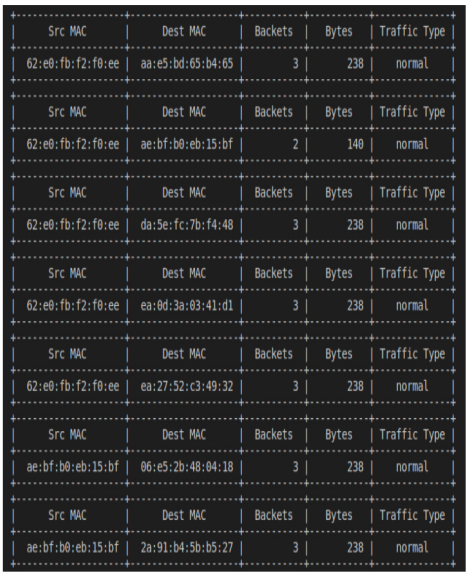
$$Y = \frac{1}{1 + e^{-x}} \tag{3}$$

5. EVALUATION AND DISCUSSION

The implementation of the DDoSDetect solution is conducted within a Mininet-simulated environment [28]. Mininet plays a pivotal role in generating the Software-Defined Networking (SDN) components, such as controllers, switches, and hosts, which can be shared across different network setups. The external controller employed in this study is the RYU controller [29]. RYU is a key component grounded in a software-defined networking framework, providing software elements with a comprehensive API that streamlines network management and control processes, enabling the identification of potential attacks. This controller is seamlessly installed via the Python PIP module's installer menu. The RYU controller then establishes a connection with the OpenFlow controller within the customized network topology simulated in Mininet.

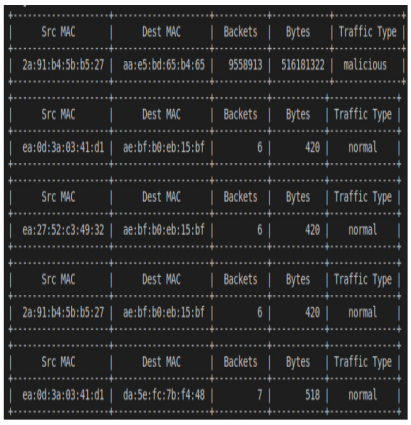
Figure 2 shows the network topology and controller-switch communication. The network consists of four switches, one controller, and eight hosts. The network addresses start from 10.0.0.1 to 10.0.0.8. Three of the hosts (10.0.0.1, 10.0.0.2, and 10.0.0.3) connected to Switch-2 and hosts (10.0.0.4, 10.0.0.5,10.0.0.6, and 10.0.0.7) connected to Switch-3. The victim server 10.0.0.8 is connected to Switch-4. The entire implementation of the DDoSDetect solution is carried out on an Ubuntu virtual machine set up on a VirtualBox machine with 4GB of RAM and 40GB of hard drive space.

To execute traffic generation, the experiment started by executing network generator script. The normal traffic are injected for three



Src MAC	Dest MAC	Backets	Bytes	Traffic Type
62:e0:fb:f2:f0:ee	aa:e5:bd:65:b4:65	3	238	normal
62:e0:fb:f2:f0:ee	ae:bf:b0:eb:15:bf	2	140	normal
62:e0:fb:f2:f0:ee	da:5e:fc:7b:f4:48	3	238	normal
62:e0:fb:f2:f0:ee	ea:0d:3a:03:41:d1	3	238	normal
62:e0:fb:f2:f0:ee	ea:27:52:c3:49:32	3	238	normal
ae:bf:b0:eb:15:bf	06:e5:2b:48:04:18	3	238	normal
ae:bf:b0:eb:15:bf	2a:91:b4:5b:b5:27	3	238	normal

Fig. 3. Detect Normal Traffic between Hosts.

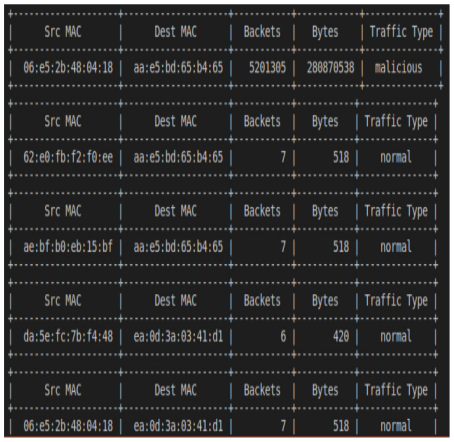


Src MAC	Dest MAC	Backets	Bytes	Traffic Type
2a:91:b4:5b:b5:27	aa:e5:bd:65:b4:65	9558913	516181322	malicious
ea:0d:3a:03:41:d1	ae:bf:b0:eb:15:bf	6	420	normal
ea:27:52:c3:49:32	ae:bf:b0:eb:15:bf	6	420	normal
2a:91:b4:5b:b5:27	ae:bf:b0:eb:15:bf	6	420	normal
ea:0d:3a:03:41:d1	da:5e:fc:7b:f4:48	7	518	normal

Fig. 4. Malicious Traffic Originating from Host Two to the Victim server.

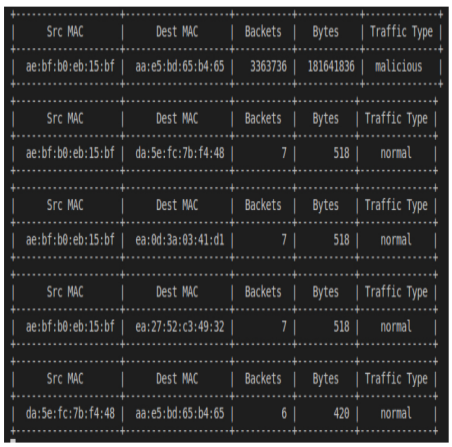
days. This includes HTTP traffic, TCP, UDP and ICMP. HTTP traffic is generated by using JupyterLab, which is an interactive web development environment. Hping3 is used for generating TCP, UDP, and ICMP traffic. The collected dataset as it is mentioned before had a total of 11,232 rows and attack traffic of 3,412 rows. The normal traffic flood from the Hosts is depicted in Figure 3. Meanwhile, Figure 4 illustrates the malicious traffic originating from Host two and targeting the victim server. Additionally, Figures 5 and 6 showcase the malicious traffic flows from Host five and Host seven, respectively, also targeting the victim server.

In order to evaluate how effectively the Logistic Regression model performs on the dataset used for training, as well as how well the Logistic Regression model performs on new, unseen data (the test set), the model's ability to accurately predict unseen data is assessed. As shown in Table 1, the training accuracy results are 97.94%, 97.79%, and 97.69% respectively with 80/20, 70/30. and



Src MAC	Dest MAC	Backets	Bytes	Traffic Type
06:e5:2b:48:04:18	aa:e5:bd:65:b4:65	5201305	280870538	malicious
62:e0:fb:f2:f0:ee	aa:e5:bd:65:b4:65	7	518	normal
ae:bf:b0:eb:15:bf	aa:e5:bd:65:b4:65	7	518	normal
da:5e:fc:7b:f4:48	ea:0d:3a:03:41:d1	6	420	normal
06:e5:2b:48:04:18	ea:0d:3a:03:41:d1	7	518	normal

Fig. 5. Malicious Traffic Originating from Host Five to the Victim Server



Src MAC	Dest MAC	Backets	Bytes	Traffic Type
ae:bf:b0:eb:15:bf	aa:e5:bd:65:b4:65	3363736	181641836	malicious
ae:bf:b0:eb:15:bf	da:5e:fc:7b:f4:48	7	518	normal
ae:bf:b0:eb:15:bf	ea:0d:3a:03:41:d1	7	518	normal
ae:bf:b0:eb:15:bf	ea:27:52:c3:49:32	7	518	normal
da:5e:fc:7b:f4:48	aa:e5:bd:65:b4:65	6	420	normal

Fig. 6. Malicious Traffic Originating from Host Seven to the Victim Server

Table 1. Training and test accuracy results

	Data Splitting Ratio		
	80/20	**70/30**	**60/40**
Training Accuracy	97.94%	97.79%	97.69%
Testing Accuracy	97.82%	97.92%	97.77%

60/40 splits. The training accuracy decreases as the training-test split ratio increases (i.e., more data is used for testing). On the other hand, the testing accuracy is highest for the 70/30 split, and slightly lower for the 80/20 and 60/40 splits. Overall observation is that the model performs well across different data splitting ratios, with training accuracy around 97.7-97.9% and testing accuracy around 97.8-97.9%. This table provides a comprehensive comparison of the model's performance under different train-test split scenarios, which is useful for evaluating the model's robustness and generalization capabilities.

On top of that, to assess how well the proposed model can make accurate predictions on new, unseen data, error rate on the test set is calculated. This allows to evaluate the model's performance outside of the training data, which is crucial for determining if the model

Table 2. Error Rates

	Data Splitting Ratio		
	80/20	**70/30**	**60/40**
Error Rates	2.18%	2.08%	2.23%

Table 3. Confusion Matrix

		Actual	
		Positive	**Negative**
Prediction	**Positive**	True Positive	False Positive
	Negative	False Negative	True Negative

can generalize well to make correct predictions in real-world scenarios. As shown in Table 2, the error rates on the unseen test data are 2.18%, 2.08%, and 2.23% for the 80/20, 70/30, and 60/40 data splitting ratios, respectively. These error rates are calculated using the method defined in Equation 4.

$$ErrorRate = 1 - Accuracy \tag{4}$$

The consistently low error rates across the different data splitting scenarios indicate that the model performs well and generalizes effectively to new data. Notably, the model achieves its lowest error rate of 2.08% with the 70/30 data split. An error rate of around 2% is generally considered a strong performance for classification tasks, meaning the model is able to accurately predict the outcomes of new, previously unseen data. This analysis demonstrates the model's ability to make reliable predictions beyond the training data, which is an important measure of its practical effectiveness and robustness. The low error rates across the different data splitting ratios highlight the model's capacity to function well and provide accurate results when applied to new information it has not encountered during the training process.

Conventionally, to evaluate the performance of the LR classification model, a confusion matrix is used - a powerful tool that provides a comprehensive view of the model's predictive capabilities. The confusion matrix is calculated based on four key parameters: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In this context, True Positive (TP) refers to correctly identified attacks, False Negative (FN) denotes attacks mistakenly classified as normal, False Positive (FP) refers to benign instances labeled as malicious, and True Negative (TN) represents accurate classification of normal instances (See Table 3). A confusion matrix allows to calculate the number of correct and incorrect prediction values for each actual class. In essence, the confusion matrix depicts the various ways in which the classification model becomes perplexed when making predictions on observations. This valuable information helps to identify the specific types of errors the model generates when assigning data to different classes.

As shown in Figure 7, Figure 8, and Figure 9 the results of confusion matrix for different data splitting ratios. These figures depict the classifier's measurement matrix, illustrating the model's performance in a traffic detection experiment assessed on testing data. With 80/20 split, TP is 637 which is the total number of correct predictions when the actual class is classified as normal, and FP is 49 which is the total number of wrong predictions when the actual class is classified as normal. In addition, TN is 1561, which is the total number of correct predictions when the actual class is classified as malicious, and FN is 0, which represents the total number of wrong predictions when the actual class is classified as malicious. With 70/30 split ratio, the TP is 930, FP is 70, TN is 2370, and FN is 0. Further, with 60/30 split ratio, the TP value is 1245, FP is 100, TN is 3148, and FN is 0.

Based on the confusion matrices values, the results indicate that the classification model is performing very well:

—The True Positive (TP) values are quite high, ranging from 637 to 1245, showing the model is correctly identifying a large number of normal instances.

—The True Negative (TN) values are also very high, ranging from 1561 to 3148, indicating the model is accurately classifying a large number of malicious instances.

—The False Positive (FP) and False Negative (FN) values are quite low, with FP ranging from 49 to 100, and FN being 0 for all data split ratios.

The low FP and FN rates, along with the high TP and TN values, suggest the classification model is performing very well and has a high degree of accuracy in distinguishing between normal and malicious instances. Overall, these results demonstrate the model has excellent performance in correctly identifying both normal and malicious cases, with very few misclassifications. This is a strong indicator that the model is capable of generalizing well and making accurate predictions on new, unseen data.

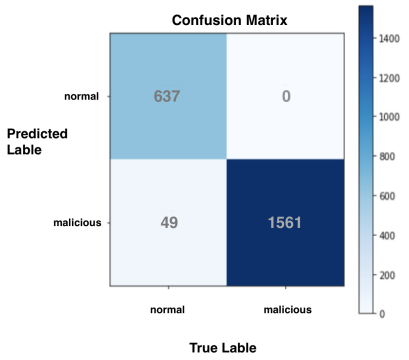


Fig. 7. Confusion Matrix of 80/20 Data Splitting Ratio.

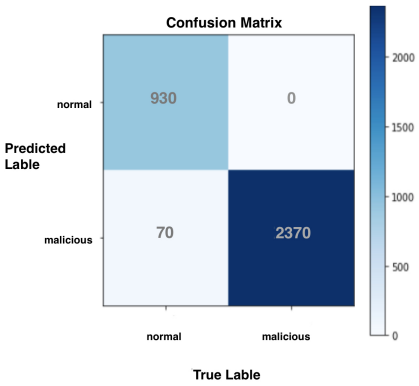


Fig. 8. Confusion Matrix of 70/30 Data Splitting Ratio.

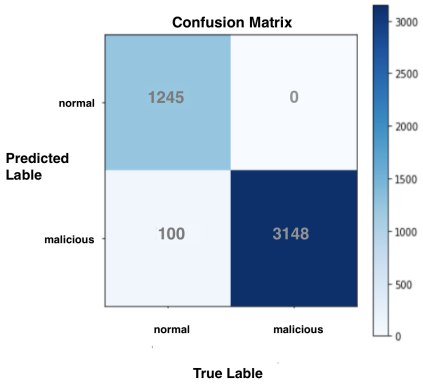


Fig. 9. Confusion Matrix of 60/40 Data Splitting Ratio.

To evaluate the efficiency of the DDoSDetect solution based on Logistic Regression algorithm, binary classification to classify the data into normal and attack traffic is performed using four classification algorithms: Naive Bayes NB, Random Forest RF, Support Vector Machine SVM, and K-Nearest Neighbor K-NN. The 70/30 data split ratio is conducted, as this configuration demonstrated the best performance with low error rates. The performance of classifier and feature selection is evaluated by using four important performance metrics, Accuracy, Precision, Recall and F-1 score. Accuracy denotes the correctness of the algorithm while detecting the attacks over the normal and the attack traffic (Equation 5). Precision denotes the percentage of positive identification of attack over total predictive positive cases (Equation 6). Recall indicates the percent of actual attack traffic that are identified correctly(Equation 7). F1 score which combines recall and precision is also computed(Equation 8).

$$Accuracy = \frac{TP + TN}{(TP + FP + TN + FN)} \tag{5}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{6}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{7}$$

$$F1Score = 2 * \frac{(Precision * Recall)}{(Precision + Recall)} \tag{8}$$

Tab1e 5 illustrates the performance of the Logistic Regression model compared to NB, SVM, RF, and K-NN. The experimental results show that the Logistic Regression model outperforms the other classifiers across several key metrics. In terms of accuracy, the Logistic Regression model achieves the highest score at 97.978%, surpassing the accuracy of NB (95.716%), SVM (97.681%), RF (95.705%), and K-NN (95.715%). Furthermore, the Logistic Regression , NB and K-NN models attain a perfect precision of 100%, indicating their ability to accurately identify positive (attack) instances without any false positives. While the RF model demonstrates the highest recall of 100%, implying its sensitivity in detecting all attack instances, the Logistic Regression model maintains a strong recall of 93.0%. Importantly, the Logistic Regression model demonstrates the best overall performance, as reflected by its superior F1-score of 96.373%, a balanced measure of precision and recall. In comparison, the F1-scores for NB, SVM, RF, k-NN are 94.748%, 94.896%, 94.479%, and 95.136%, respectively. These findings clearly illustrate the effectiveness of the Logistic Regression-based DDoSDetect solution in accurately and reliably detecting DDoS attacks in the context of software-defined networking environments.

Table 4. DDoS detection classifier performance metrics in an SDN environment.

	Performance Metrics			
Classifier Algorithms	**Accuracy%**	**Precision%**	**Recall%**	**F1-Score%**
Logistic Regression	97.978	100	93	96.373
Naive Bayes	95.716	100	90.021	94.748
Support Vector Machine	97.681	99	91.12	95.896
Random Forest	95.705	98.537	100	94.479
K-NN	95.715	100	90.723	95.136

The evaluation of the Logistic Regression model reveals its effectiveness and adaptability. By examining training and testing accuracies across various data splits, the experiment provides insights into the model's behavior with different data amounts. Low error rates on test data indicate the model's stability and reliability, even with varying data proportions. Achieving a 2.08% error rate on a 70/30 split demonstrates strong predictive power. This accuracy is crucial for detecting network attacks effectively. Overall, the model effectively distinguishes between normal and malicious traffic, showing reliability and potential for enhancing network security against DDoS attacks. The model's superior metrics and consistent performance demonstrate its dependability and potential as a useful tool for strengthening network security and successfully fending off potential cyber threats.

6. CONCLUSIONS AND FUTURE WORK

The evolution of network technologies has necessitated a shift towards Software-Defined Networking (SDN) to address the limitations of traditional network architectures. SDN brings advantages like centralized control, compatibility with various vendors, and improved flexibility, making operations more cost-effective and efficient. While the SDN market is expected to grow, worries linger about security risks, especially concerning Distributed Denial of Service (DDoS) attacks. SDN environments are vulnerable to DDoS attacks, which target several network architecture layers and necessitate a thorough security plan for prevention. Because SDN is programmable, it offers opportunities for dynamic mitigation through machine learning-based defense mechanisms and flow rule adjustments, whereas conventional networks used resource-intensive techniques to counter DDoS attacks.

This paper introduced DDoSDetect, employing Logistic Regression to detect DDoS attacks in SDN environments, emphasizing flood-based attacks like TCP SYN, HTTP, UDP, and ICMP. The solution analyzes SDN traffic using key features like packet count, size, and MAC addresses to differentiate between normal and attack traffic. Evaluation of the proposed model against Naive Bayes, Random Forest, K-NN and SVM classifiers reveal that DDoSDetect, leveraging Logistic Regression, outperforms alternatives with a 2.4% accuracy boost, a 2.0% F1-score improvement, and an 11.68% precision increase, showcasing its effectiveness in combating DDoS threats in SDN networks.

Future work includes implementing mitigation strategies upon DDoS detection in SDN, along with exploring larger datasets and conducting thorough feature selection analysis. Furthermore, an upcoming focus will involve the development of a hypermodel that

incorporates multiple classifier algorithms to improve detection accuracy and robustness against DDoS attacks in SDN environments.

7. ACKNOWLEDGEMENTS

We would like to show our appreciation to the following master students who participated in applying this study successfully: Haneen Abdulqader Shaer, and Maha Abdullah Alqarni, Faculty of Computing and Information Technology, King Abdulaziz University. The authors also would like to express their gratitude to the AI language model ChatGPT, for its invaluable assistance in the writing process of this paper.

8. REFERENCES

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[2] W. Braun and M. Menth, "Software-defined networking using openflow: Protocols, applications and architectural design choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, 2014.

[3] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[4] "Marketsandmarkets, 2019, software defined networking market." https://https://www.marketsandmarkets.com/Market-Reports/, [Online; accessed 30-July-2024].

[5] "Oleg kupreev, ekaterina badovskaya, a. g., 2020. ddos attacks in q1 2020," https://securelist.com/ddos-attacks-in-q1-2020/96837/, [Online; accessed 6-June-2024].

[6] A. N. Viet, L. P. Van, H.-A. N. Minh, H. D. Xuan, N. P. Ngoc, and T. N. Huu, "Mitigating http get flooding attacks in sdn using netfpga-based openflow switch," in *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2017, pp. 660–663.

[7] S. Wang, K. G. Chavez, and S. Kandeepan, "Seco: Sdn secure controller algorithm for detecting and defending denial of service attacks," in *2017 5th International Conference on Information and Communication Technology (ICoIC7)*. IEEE, 2017, pp. 1–6.

[8] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research trends in security and ddos in sdn," *Security and Communication Networks*, vol. 9, no. 18, pp. 6386–6411, 2016.

[9] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014.

[10] "Sflow-rt," https://sflow-rt.com, [Online; accessed 25-June-2024].

[11] K. Kalkan, L. Altay, G. Gür, and F. Alagöz, "Jess: Joint entropy-based ddos defense scheme in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, 2018.

[12] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A ddos attack detection method based on svm in software defined network," *Secur. Commun. Networks*, vol. 2018, pp. 9 804 061:1–9 804 061:8, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:21717772

[13] L. Yang and H. Zhao, "Ddos attack identification and defense using sdn based on machine learning method," *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, pp. 174–178, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:59619582

[14] J. Kang, Y. Zhang, and J.-B. Ju, "Classifying ddos attacks by hierarchical clustering based on similarity," in *2006 International Conference on Machine Learning and Cybernetics*. IEEE, 2006, pp. 2712–2717.

[15] L. Barki, A. Shidling, N. Meti, D. Narayan, and M. M. Mulla, "Detection of distributed denial of service attacks in software defined networks," in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2016, pp. 2576–2581.

[16] S. Dong and M. Sarem, "Ddos attack detection method based on improved knn with the degree of ddos attack in software-defined networks," *IEEE Access*, vol. 8, pp. 5039–5048, 2019.

[17] S. Yadav and S. Selvakumar, "Detection of application layer ddos attack by modeling user behavior using logistic regression," in *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*. IEEE, 2015, pp. 1–6.

[18] N. Meti, D. Narayan, and V. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," in *2017 international conference on advances in computing, communications and informatics (ICACCI)*. IEEE, 2017, pp. 1366–1371.

[19] K. S. Sahoo, A. Iqbal, P. Maiti, and B. Sahoo, "A machine learning approach for predicting ddos traffic in software defined networks," in *2018 International Conference on Information Technology (ICIT)*. IEEE, 2018, pp. 199–203.

[20] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A ddos attack detection method based on svm in software defined network," *Security and Communication Networks*, vol. 2018, no. 1, p. 9804061, 2018.

[21] N. Z. Bawany, J. A. Shamsi, and K. Salah, "Ddos attack detection and mitigation using sdn: methods, practices, and solutions," *Arabian Journal for Science and Engineering*, vol. 42, pp. 425–441, 2017.

[22] M. W. Nadeem, H. G. Goh, V. Ponnusamy, and Y. Aun, "Ddos detection in sdn using machine learning techniques." *Computers, Materials & Continua*, vol. 71, no. 1, 2022.

[23] J. Bhayo, S. A. Shah, S. Hameed, A. Ahmed, J. Nasir, and D. Draheim, "Towards a machine learning-based framework for ddos attack detection in software-defined iot (sd-iot) networks," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106432, 2023.

[24] H. Alubaidan, R. Alzaher, M. AlQhatani, and R. Mohammed, "Ddos detection in software-defined network (sdn) using machine learning," *Int J Cybernetics Inform*, vol. 12, no. 4, 2023.

[25] H. Elubeyd and D. Yiltas-Kaplan, "Hybrid deep learning approach for automatic dos/ddos attacks detection in software-defined networks," *Applied Sciences*, vol. 13, no. 6, p. 3828, 2023.

[26] "jupyter," https://jupyter.org/, [Online; accessed 25-June-2023].

[27] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *2016 IEEE 24th International conference on network protocols (ICNP)*. IEEE, 2016, pp. 1–5.

[28] "Mininet," http://mininet.org, [Online; accessed 25-June-2023].

[29] "jupyter," https://ryu.readthedocs.io/en/latest/index.html, [Online; accessed 20-Jan-2023].