International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V71I2P237
Volume 71 Issue 2, 349-361, February 2023

Original Article

XGBoost Machine Learning Model-Based DDoS Attack Detection and Mitigation in an SDN Environment

Arvind T[1], K. Radhika[2]

[1]*Department of CSE, UCE, OU, Hyderabad, Telangana, India.*
[2]*Department of IT, CBIT, Telangana, India.*

[1]*Corresponding Author : mr.arvind@rediffmail.com*

Received: 19 November 2022 Revised: 04 February 2023 Accepted: 18 February 2023 Published: 25 February 2023

Abstract *- SDN sparked tremendous interest because of its several benefits, such as simple programming, quick scalability, centralized administration, etc. However, security is a significant problem, and Distributed denial of service (DDoS) threats a major challenge for SDN. One way to safeguard a Software-Defined networking infrastructure from DDoS assaults is to use machine learning models. This study presents an XGBoost-based approach for DDoS detection and mitigation. It evaluates it against other Machine Learning techniques, including Logistic Regression, Naive Bayes, Decision Trees, XGBoost, and Multilayer Perceptron. This method will generate, collect, classify, detect, and then mitigate Distributed denial-of-service assaults. The results show that the suggested approach protects SDN from DDoS attacks with high accuracy and a low error level while making good use of network resources. Despite the short training and testing period, the proposed method detects DDoS attacks with greater accuracy.*

Keywords *- SDN, DDoS, Machine learning, Mininet, Ryu.*

1. Introduction

In this dynamic era, conventional networks face a lot of difficulties, such as vendor dependency, lack of support for dynamic policy updates, etc. SDN overcomes these difficulties by transferring all the decision-making capabilities to the control plane and reducing the data plane to the role of a simple packet forwarding unit. This capability of the SDN made it widely spread. However, at the same time, it also made it a primary cause of its failure. One of the popular attacks that can happen on the SDN controller is DDOS, where the controller is overwhelmed with a huge volume of packets from multiple distributed hosts. The switch's flow table also becomes full due to incoming data packets from multiple distributed sources, leading to reduced packet forwarding and dropping of incoming packets. These DDoS attacks can be of various types, such as volume-dependent, protocol-dependent, and application-dependent. Volume-dependent assaults include flooding attacks such as ICMP, UDP, and so on. Protocol-dependent assaults, such as SYN floods, the Ping of Death, Smurf attacks, and so on, target server resources. Active apps in the application plane are the target of application-dependent assaults. Slowloris and zero-day attacks are among them. Several approaches exist for detecting and defending against DDoS assaults; however, Machine learning (ML) techniques offer a promising solution for early detection due to their faster response times compared to manual methods. SDN flow data can be processed using ML-based DDoS attack detection systems integrated into SDN topologies to create an autonomous, adaptable network. However, the current state of research lacks SDN benchmark datasets and a model that can effectively and accurately predict DDoS attacks using ML techniques. In light of this, there exists a significant research gap in the area of effectively detecting and preventing DDoS attacks in SDN. This study proposes an XGBoost-based DDoS detection and defense system. The procedure includes traffic generation, collection, classification, detection, and mitigation.

The paper includes the following sections: Section 2 explains the effects of DDoS on SDN networks, Section 3 explores the related studies, Section 4 describes the proposed system's experimental settings, Section 5 explores the modules of the proposed approach, and Section 6 discusses the Conclusion and Future Considerations.

2. DDoS attacks in SDN

SDN (Software-Defined Networking) has features that can make it more resilient to DDoS (Distributed denial of service) attacks, as well as features that can be used to victimize others with DDoS attacks.

The features such as centralized management, programmability, traffic engineering, and dynamic flow management make it more resilient to DDoS attacks. While it

can be vulnerable to distributed denial of service (DDoS) attacks as it has features that attackers can exploit, these victimizing features include centralized control, security risks, performance concerns, and a lack of standardization. The DDoS attacks target different layers of SDN.

- Control plane DDoS attacks include flooding, resource exhaustion, poison packet attacks, and so on.
- Data-plane DDoS attacks: can take the form of TCAM exhaustion, Flooding, Data-Control plane link saturation, etc.
- Application-plane DDoS attacks: can take the form of flood attacks, application-layer attacks, command injection attacks, resource exhaustion attacks, and configuration tampering attacks.

3. Related Work

This section of the article focuses on exploring the studies that employ statistical, machine learning, and deep learning techniques.

3.1. Statistical-Based Techniques

Statistical analysis uses statistical methods to look for strange patterns in network traffic that could be signs of a DDoS attack. Methods like mean, median, mode, standard deviation, entropy, and chi-square analysis can be used to find outliers in network data that could be signs of an attack [1,2].

Giotis et al. [4] devised an entropy technique to collect and analyze data to identify network anomalies, thereby reducing the controller's workload. They tested their system on the "National Technical University of Athens" network, where they gathered regular traffic for the anomaly detection module. This module analyses all flow inputs across all time intervals to determine undesirable ones. The mitigation module then specifies a flow rule to block the originating IP. With the assistance of Tcpreplay and Scapy, they were utilized to produce malicious traffic.

Mousavi et al. [5] presented a strategy to identify DDoS assaults using Shannon's entropy. All nodes will experience the same amount of traffic and entropy while the network is running properly.[6] When one or more hosts are exposed to a DDoS attack, they experience a fall in entropy due to an unusually large amount of traffic. After receiving 50 packet-in messages, the controller estimates the entropy using the target IP address. An attack is recognized when the estimated entropy falls below the detection threshold for five consecutive rounds.

The authors, Leu et al. [7], proposed agent-based IDS using the goodness of fit test of the Chi-Square to identify DoS and DDoS assualts. It examines the number of SrcIP variations that send packets to the target and Ipaddr distribution statistics. If an attack is detected, the chi-square value exceeds the threshold.

By integrating entropy-based techniques with machine learning algorithms, Dehkordi et al. [13] were able to identify low and high volumes of DDoS assaults. The method involves determining when detection is optimal in order to maximize efficiency. The suggested solution outperforms previous DDoS protection algorithms in terms of accuracy (99.85%).

Mishra et al. [9] demonstrated an entropy-based DDoS defense system with minimum computing costs. The suggested technique makes use of three different threshold values: entropy, packet flow rate, and a count threshold. Once the predetermined threshold is exceeded, the entropy is computed by a controller using data from the flow table of the associated switches. The count value is increased when the calculated entropy drops below the predetermined threshold. When the overall number of assaults hits a specified threshold, an alert is triggered. As part of the mitigation procedure, a controller will collect the offending IP address, DPID, and switch port information and begin discarding packets from that IP address immediately.

3.2. Machine Learning-Based Techniques

ML algorithms, like Decision Trees, Naive Bayes Random Forest, and Neural Networks, can be trained on normal network traffic patterns and used to detect anomalies that may indicate a DDoS attack. These algorithms can also be used to differentiate between benign and malicious traffic based on characteristics such as source IP address, destination IP address, packet size, and packet rate.

Saurav et al. [32] studied the attack patterns in the network by utilizing ML techniques. The methodology uses 4 different ML algorithms: C4.5, Naive Bayes, Bayes Net, and Decision Table. The models' prediction accuracy was examined, and they concluded that the Bayesian network had the greatest prediction rate.

Santos et al. [18] used a Mininet emulator and POX controller to produce the DDoS dataset with 23 characteristics. The traffic was generated using Scapy. They used different ML models to analyze the dataset, including SVM, MLP, DT, and RF. The findings indicate that the DT has a shorter processing time, and the RF model has the highest accuracy.

Swami et al. [21] presented an ML-based intrusion detection system to identify TCP-SYN flooding assaults. The authors created the traffic as well as classified and analyzed the performance of many models, including DT, LR, RF, MLP, and Adaboost. The experimental research made use of Mininet, the Ryu controller, and the Scapy tool to create traffic, the Tcpdump tool to collect traffic, and the Wireshark tool to analyze packets.

Obaid et al. [17] employed a range of machine learning models, including RF, KNN, J48, and SVM, to detect and mitigate DDoS attacks in an SDN network. Weka was used to train and test the models. Ubuntu served as the platform for

the experiment, while Mininet and Ryu controllers were used to facilitate the process. They used Tshark to produce normal traffic, while Hping3 was used to produce malicious traffic. The findings of the tests provided support for the J48 model as the most promising model.

StateSec is an entropy-based DDoS detection technique that operates on the data plane and is developed by Rebecchi et al. [22]. This was done for the controller's resource conservation. StateSec's monitoring and detection mechanisms are constructed using an OpenFlow switch. The switch lets the controller know when an attack is happening, so the controller can take defensive action. The findings indicate a considerable reduction in controller overhead and a high level of detection accuracy.

Nisharani et al. [15] investigated the effectiveness of three machine learning algorithms: SVMs, Naive Bayes, and Neural Networks. Mininet and Ryu were critical components of the experimental setup. Compared to the other models, the SVM achieved the greatest accuracy, recall, and precision levels.

A hybrid ML approach was suggested by Trung et al. [26] to enhance the classification of traffic using SVM and SOM approaches. Additionally, an improved "History-Based IP Filtering system" (eHIPF) was developed to rapidly and correctly discover attacks. They were able to reproduce real-world situations in a cloud environment that simulates a software-defined network using a service function chaining technique. The experimental findings revealed that the proposed strategy outperformed the other alternatives considered.

Tan et al. [11] devised a DDoS detection trigger mechanism for SDN switches to lower control-plane overhead. When suspect data flows are discovered, data plane switches notify the controller. In response to the alert, the controller implemented a hybrid ML-based DDoS detection approach based on KNN and K-means. This method saves the controller's resources while maintaining a high degree of accuracy (98.85 percent).

Ahuja et al. [10] created an SDN-based dataset with attributes that could be used to detect fraudulent traffic. Open-Flow switch port and flow data are employed when constructing the dataset. An SVC-RF-based hybrid machine-learning technique is applied to distinguish between benign and malicious packets. The suggested approach has a low false-positive rate (0.020) and high precision (98.8%).

3.3. Deep Learning-Based Techniques

Deep learning methodologies, like CNNs and RNNs, can be utilized to analyze network traffic in real-time and detect DDoS attacks. These algorithms can learn complex patterns and relationships in network traffic, making them well-suited for detecting sophisticated DDoS attacks.

Tsung [45] et al. suggested DL-IDPS, a deep learning-based system to protect the SDN from brute-force and DDoS attacks. To find brute-force and DDoS attacks, the MLP, CNN, LSTM, and SAE deep learning models were made and compared. The MLP model did better than previous ones. It predicted SSH brute-force attacks with a 99.9% accuracy rate and a prediction time of 38.4 msec. It also predicted DDoS attacks with a 98.3% accuracy rate and a prediction time of 27.2 msec. The experiment was carried out on Ubuntu Linux using VMware and a Ryu controller. SSH brute-force attacks were carried out using the Ncrack and Medusa tools, while DDoS assaults were carried out using the Hping3 tool. Flow-based characteristics such as src IP, dst IP, src port, dst port, and so on were extracted.

A system based on LSTM and fuzzy logic was suggested by Novaes et al. [33] to identify and prevent DDoS and port scan assaults. Synthetic datasets and CICDoS 2019 datasets were used to assess performance in Mininet and Floodlight controller environments. The data gathered proves the usefulness of the suggested solution.

To identify DDoS assaults, Nugraha et al. [8] used a hybrid CNN-LSTM technique. The model's efficacy was assessed using data that had been generated synthetically. The experimental evaluation revealed that the proposed model outperformed the MLP and the 1-class SVM models.

3.4. Other Techniques

Other methods, such as Fuzzy logic, Bloom filter, Genetic algorithm, etc., can be used to detect DDoS attacks.

Zohaib et al. [34] suggested a DoS and DDoS detection system based on SNORT. Snort profiles with DDoS flow rules were created to detect and prevent DDoS assaults; however, the suggested method consistently produced false alarms.

The authors proposed a technique based on the Bloom filter to identify link flooding attacks on SDN. The implementation was performed on a Mininet, utilizing the Floodlight controller and Java-based JSON and JPCAP APIs. Iperf was used to block the links and simulate a link flood attack, and five characteristics were collected from the generated traffic: src IP, dst IP, src port, dst port, and protocol. The authors ran numerous trials with numerous datasets, configurations, detection times, and false-positive metrics.

The authors [41] suggest Adaptive Bubble Burst (ABB), a unique mitigation approach, to supplement classic packet filtering and other DDoS defenses. Adaptive Bubble Burst helps to preserve service availability for a targeted resource in the face of a distributed denial of service (DDoS) assault. ABB provides two separate safety choices. It can secure your privacy and avoid Distributed Denial-of-Service assaults. ABB conceals the real nature of a protected service by promoting various services from a single virtual IP address.

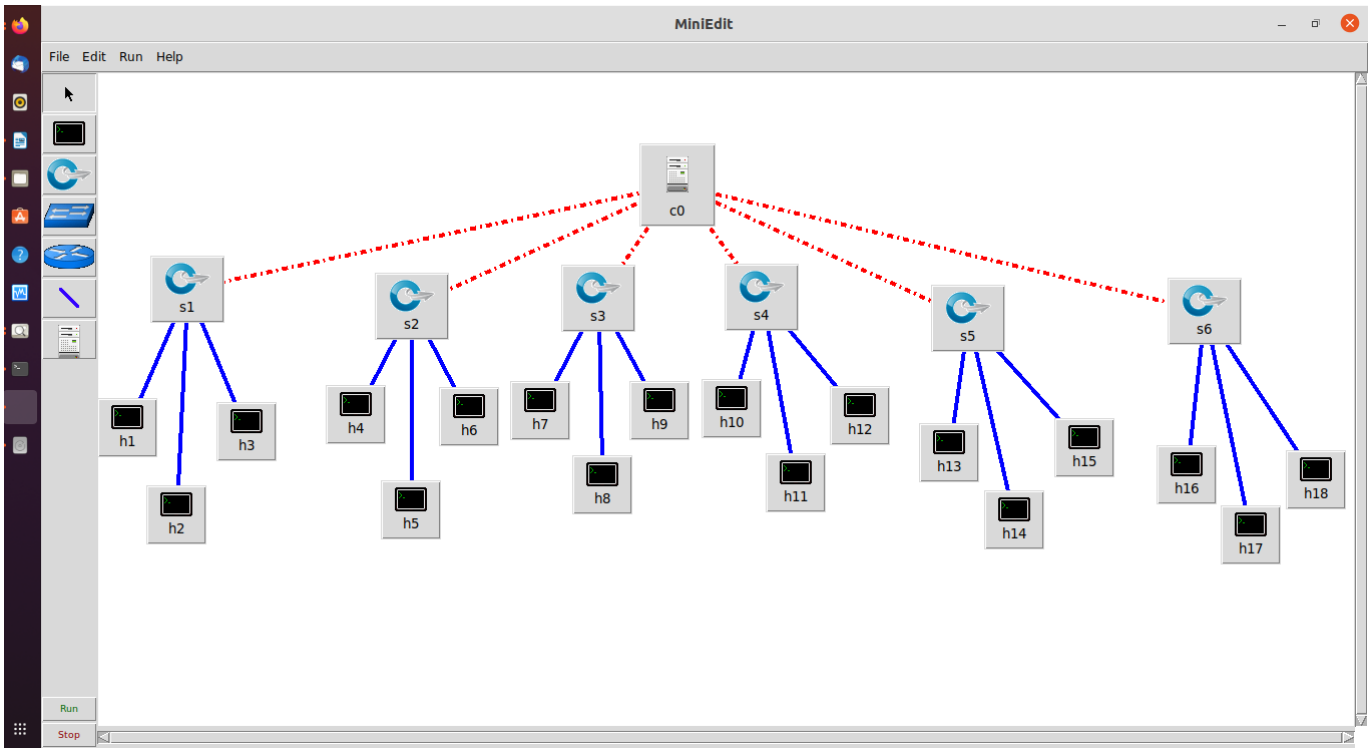

Fig. 1 Mininet Emulated Topology

Second, if a DDoS attack bubble is heading toward a resource, ABB will mitigate it by adaptively dispersing it over numerous copies of that resource in the network. As a result, the primary resource suffers less damage. ABB accomplishes nothing in terms of detecting or avoiding threats. To the best of my knowledge, ABB offers no protection against DDoS attacks. Despite the fact that attack traffic will make resources more accessible and continue to circulate around the network, ABB will make the resource more accessible.

4. Experimental Setup

This section defines the testbed as well as the topology of the experiment.

4.1. Experimental Environment and Tools

Experiments were carried out on a system running the 64-bit version of Ubuntu 20.04 LTS, with Mininet [31], Python-based opensource OpenFlow SDN controller Ryu [28.29], Python and Jupyter [24] notebook installed. Ping and Iperf were utilized to produce benign traffic, while the Hping3 application was utilized to produce attack traffic [1]. A Mininet emulator was used to build the network architecture for the proposed system, which has six switches (OVS) and 21 hosts, as shown in Fig.1. A dedicated control channel links the switches to the controller to facilitate communication between the two.

5. Modules of the Proposed System

The proposed system consists of four different modules, which are as follows:

1. Traffic Data Generation and Collection
2. Machine learning classification of traffic
3. Implementation of Detection framework
4. Implementation of the Mitigation framework

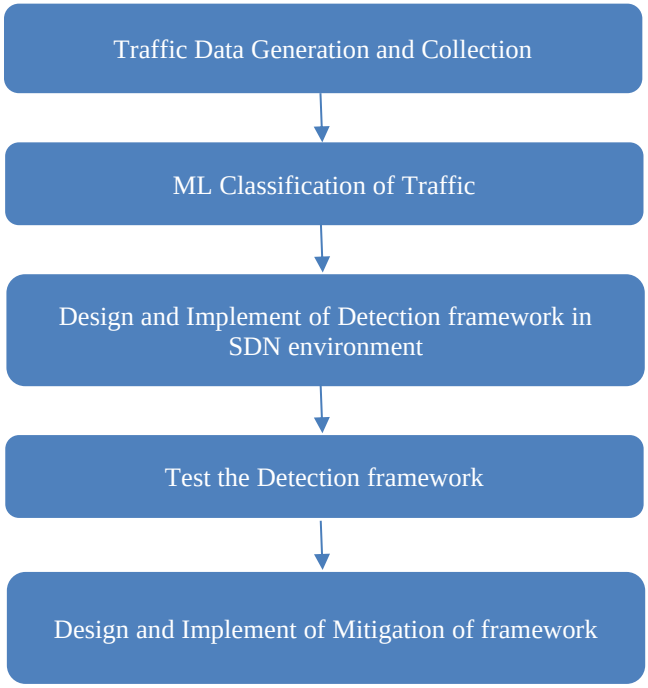

Fig. 2 Modules of the System

5.1. Traffic Data Generation and collection

The technique includes generating both benign and malicious traffic. It produced three types of traffic—normal and attack traffic—over TCP, UDP, and ICMP and saved each type in a separate CSV file. Ping and Iperf were utilized to create normal traffic, while the Hping3 utility was utilized to simulate malicious traffic.

To get a decent balance of normal and malicious traffic, data on normal traffic was collected for about 40 hours, and data on malicious traffic was collected for about 90 minutes [1, 16–20]. Following traffic collection, a shell script was used to include a target column in both the attack and normal datasets. Under Linux, the awk function is used to add the specified column to the output. The datasets were combined using a script that makes use of the Cat programme. Finally, the dataset was randomized using a script that makes use of the Linux shuf utility. There are twenty-four attributes in the data, including datpath_id, timestamp, flow_id, src_ip, dst_ip, tp_src, tp_dst, ip_proto, icmp_code, icmp_type, flow_dur_sec, flow_dur_nsec, idle_timeout, hard_timeout, flags, in_port, byte_count. The dataset was then preprocessed and split into training and testing groups, after which machine learning classifiers were trained and tested to evaluate their performance.

Table 1. Traffic Distribution of the Dataset

Traffic	ICMP	UDP	TCP	Total
Normal	206101	121200	177714	505015
Attack	152353	143795	165547	461695
Total	358454	264995	343261	966710

5.2. Machine Learning Classification Of Traffic

Several different Machine Learning models, namely Logistic Regression, Naive Bayes, XGBoost, Decision Tree, and Multilayer Perceptron, were studied in order to identify the traffic that was present in the dataset.

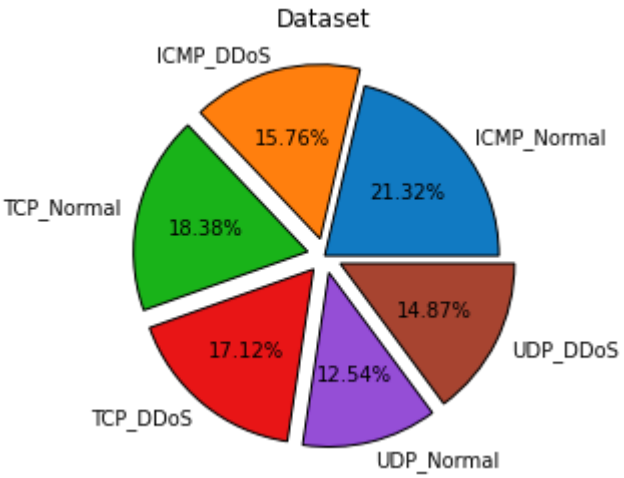

Fig. 3 Dataset Traffic Distribution

5.2.1. Machine Learning DDoS Classification Algorithm

Input: Network traffic, both legitimate and malicious.
Output: optimal classifier for DDoS detection

Begin

Step 1: Data Collection
Collect the network traffic; this includes features that can distinguish between normal traffic and attack traffic.

Step 2: Preprocess the data
Clean and preprocess the collected data to make it suitable for the classifier.

Step 3: Split data into training and testing groups.
Divide the data into two groups: one for training the classifier and the other for testing its performance.

Step 4: Train the classifier.
Train the classifier on the training data.

Step 5: Test the classifier.
Use the trained classifier to make predictions on the test data.

Step 6: Evaluate the performance of the classifier.
Evaluate the classifier's performance by comparing its predictions with the labels in the test data.
Repeat steps 1–6 for all the classifiers.

Step 7: Compare and select the best classifier.
Compare the performances of the classifiers and select the best one for DDoS detection.

Step 8: Implement the classifier for real-time traffic detection.

End

5.2.2 Evaluation Metrics

Several distinct metrics, including accuracy, recall, precision, f1-score, training time, testing time, and ROC-AUC, were used to assess the models [1]. TP is used to indicate the occurrence of positive events that are properly identified as positive outputs, while FP is used to indicate the occurrence of negative events that are wrongly classified as positive outputs. FN stands for positive occurrences that have been incorrectly classified as negative output. TN stands for negative occurrences that have been correctly labeled as negative.

Accuracy: It is defined as the degree to which a measured value comes close to the actual (true) value. It is possible to express it as the percentage of properly classified occurrences relative to the total number of occurrences. It can be given as

(True Positive + True Negative)/Total number of occurrences (1)

Precision: The term "precision" refers to how close the values that were measured are to one another. It can be given as

True Positive / (True Positive + False Positive) (2)

Recall: or Sensitivity, can be given as

True Positive / (True Positive + False Negative) (3)

F1-score: is the harmonic mean of Precision and Recall.

2* (Precision * Recall / (Precision + Recall)) (4)

Table 2. Performance Comparison

	Accuracy	Precision	Recall	F1
NBs	0.817828	1.000000	0.618626	0.764384
LR	0.522327	0.000000	0.000000	0.000000
XGB	0.99999	1.000000	1.000000	1.000000
DT	0.9999	1.000000	1.000000	1.000000
MLP	0.540033	0.509442	0.999965	0.674999

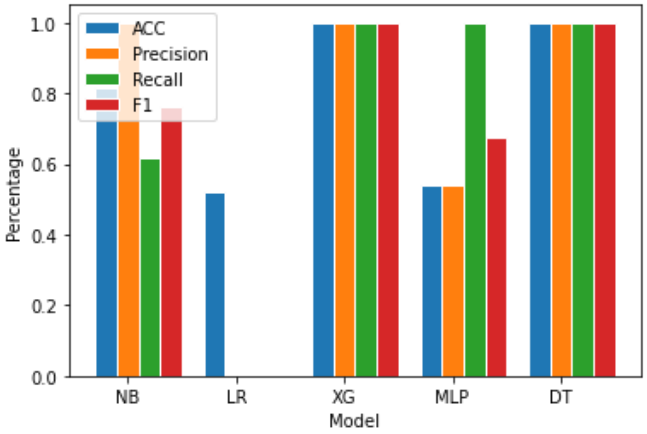

Fig. 4 Evaluation Metric Comparison

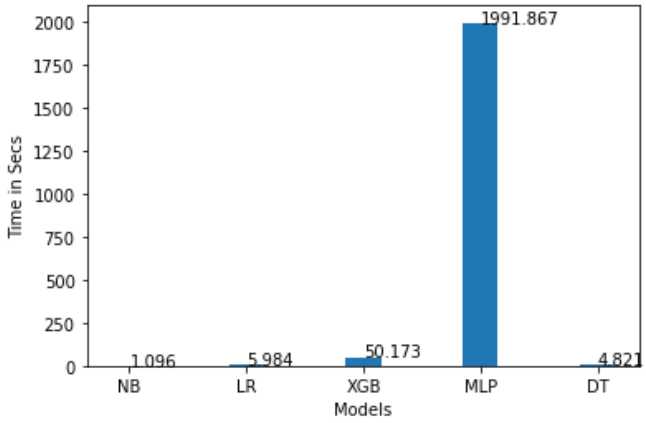

Fig. 5 Training time Comparison

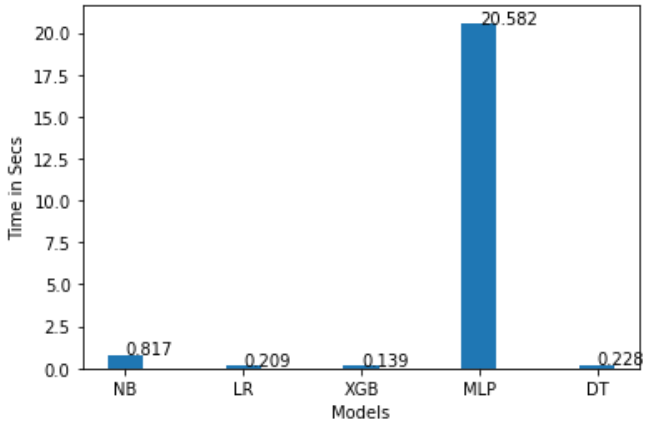

Fig. 6 Testing Time Comparison

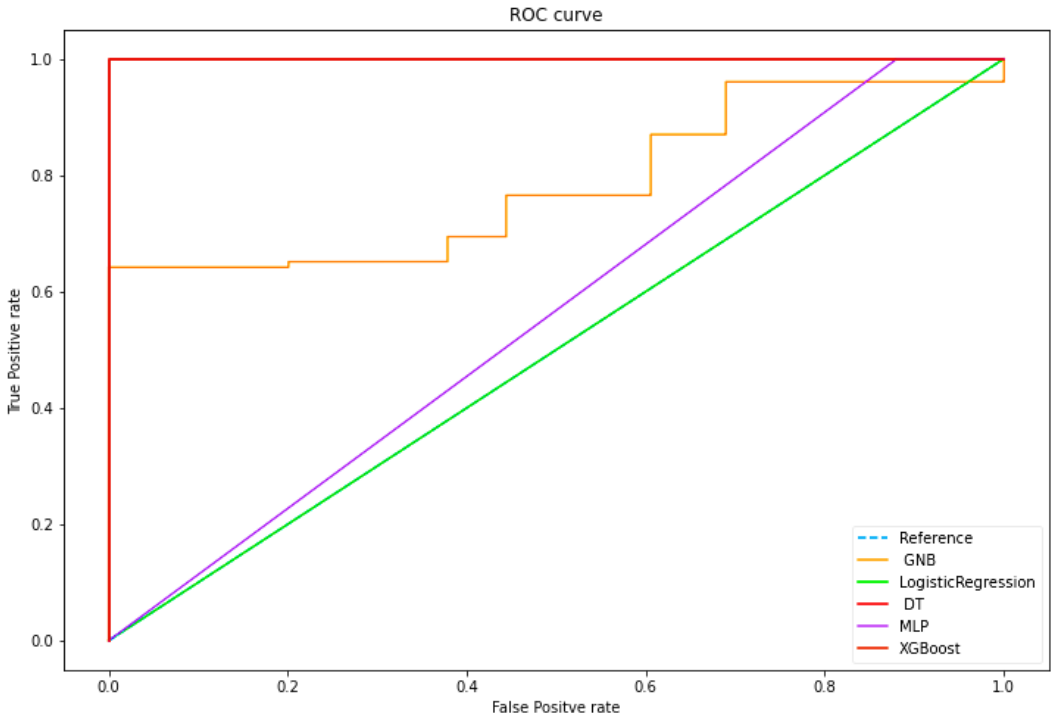

Fig. 7 AUC-ROC Comparison

Fig. 4 illustrates the detection accuracy of the models, and the table illustrates the measures of Accuracy, Precision, Recall, and F1-scores of various classifiers. The evaluated results suggest that XGBoost and Decision Tree classifiers provide better accuracy than other classifiers. Gaussian Naive Bayes, XGBoost, and Decision Tree classifiers provide better Precision than Logistic Regression and Multilayer Perceptron classifiers. The Recall of the Decision Tree, MLP, and XGBoost classifiers were almost the same. Furthermore, the F1 score provided by the XGBoost and Decision Tree classifiers is higher when compared to the scores provided by the other classifiers.

The graph in Fig. 5 shows a bar graph that depicts the training times needed by the classifiers. The graph's X-axis indicates training time in seconds, while the Y-axis represents classifiers. In this evaluation, it can be seen that the training time for the Multilayer Perceptron model is much higher than that of the other classifiers. The time needed to train the Naive Bayes algorithm is much shorter than that required by other models. The DT model's training time is lower than the XGB and MLP models.

The graph in Fig. 6 shows a bar graph that depicts the amount of time spent testing the models. The graph's X-axis indicates the testing time in seconds, while the Y-axis represents classifiers. The findings demonstrate that the testing time of XGBoost is less than that of any other model. The MLP model requires more testing time than other models. The DT model requires more testing time than the XGBoost, and LR models.

The graph in Fig. 7 represents the Receiver Operating Characteristic (ROC) curve, which is constructed to evaluate test results for the model. The area under the ROC is a two-dimensional graph that shows the FPR and TPR on the X and Y axes, respectively. Compared to other models, XGBoost and DT have the greatest AUC (0.999), whereas Gaussian Naive Bayes, MLP, and Logistic Regression have 0.785, 0.559, and 0.500, respectively.

5.3. Implementation of Detection Framework

Monitoring the network is a necessary part of detecting the detection process, and in this step, the best classifier model from the earlier phase must be chosen. The XGBoost model and the Decision Tree model outperformed the other models in the previous phase, so we built an XGBoost-based detection model. The dataset was initially used to train the detection model, which was then used to identify the traffic that was being generated. The data were first collected in CSV format for the purpose of identifying traffic. After that, the data were preprocessed, and then they were fed into the model, which subsequently made a prediction about whether or not the traffic was legitimate or malicious.

5.3.1. XGBoost-Based DDoS Detection Algorithm

XGBoost is a machine learning algorithm that uses gradient boosting for classification and regression tasks. It improves the model's accuracy by fitting new trees to the residuals (errors) of the current prediction. The new trees are optimized by finding the best-split points and feature values to minimize the loss (error) of the model. This cycle is maintained until some stopping condition is satisfied, such as a predetermined maximum number of trees or a certain minimum improvement in the loss function.

Finally, all the trees are combined to form the final prediction model. XGBoost includes features like regularization to avoid overfitting, parallel processing for faster computation, and handling missing data.

Algorithm: DDoS Detection and Mitigation

Input: Network traffic, both legitimate and malicious.

Output: Optimal XGBoost classifier

Begin

Step 1: Data Collection

Collect the network traffic; this includes features that can distinguish between normal traffic and attack traffic.

Step 2: Preprocess the data

Clean and preprocess the data to prepare it for use in the classifier.

Step 3: Split data into training and testing groups.

Divide the data into two groups: one for training the classifier and the other for testing its performance.

Step 3: Train and Validate the Model

Train and validate the model with the data when the controller starts.

Step 4: Monitor and analyze traffic

Preprocess and analyze the monitored traffic to identify any potential DDoS attacks.

Step 5: Detect the attack.

Detect and confirm the presence of a DDoS attack based on the analysis results.

If traffic is determined to be attacked, traffic trigger the mitigation module to block the switch input port

Else

Forward the traffic

Repeat steps 4 and 5 for all incoming traffic.

End

5.3.2. Detection of Normal and Attack traffics

Fig.8 depicts typical normal traffic detection, which requires running the Mininet topology on one terminal and the XGBoost-based detection program on another, then accessing the terminal windows of several hosts, such as hosts h1, h10, using the xterm command under Mininet. Normal traffic was then switched from h10 to h1.

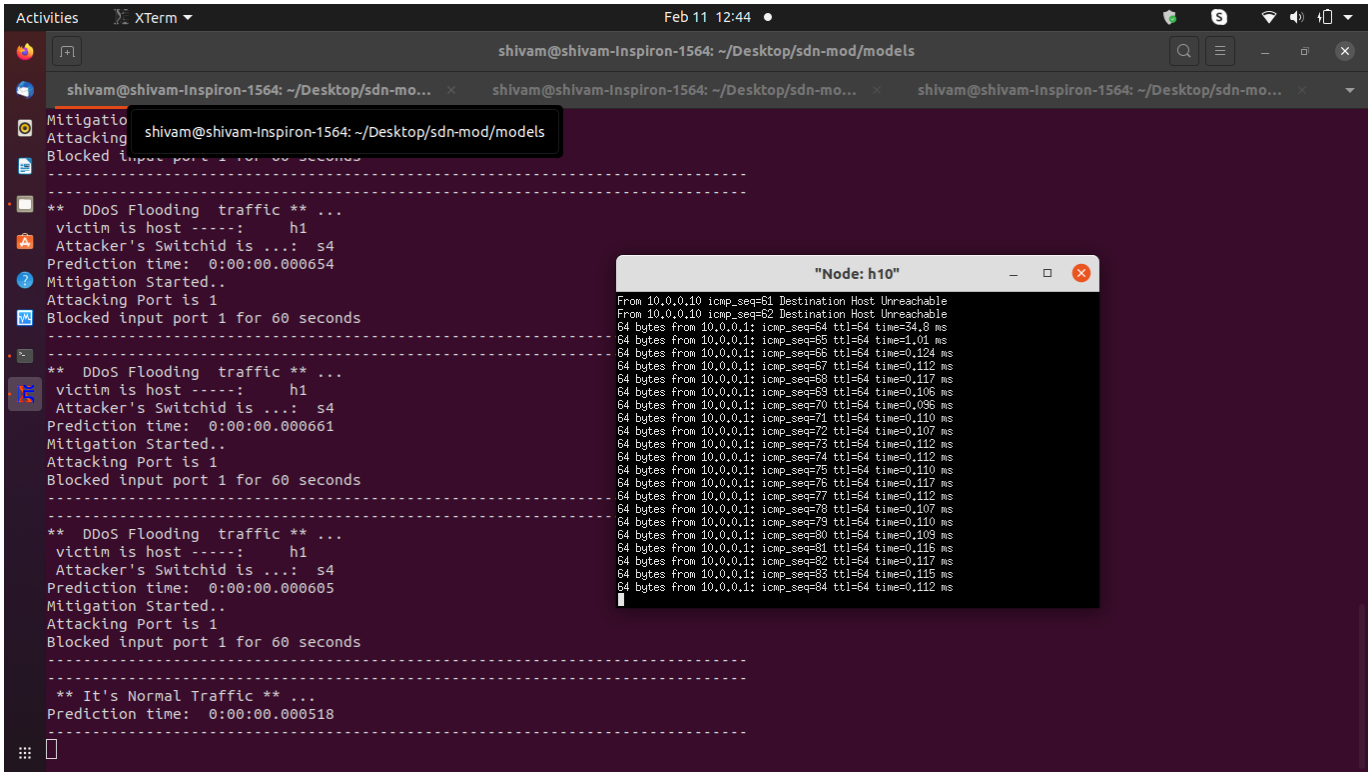

Fig. 8 Normal-Traffic Detection

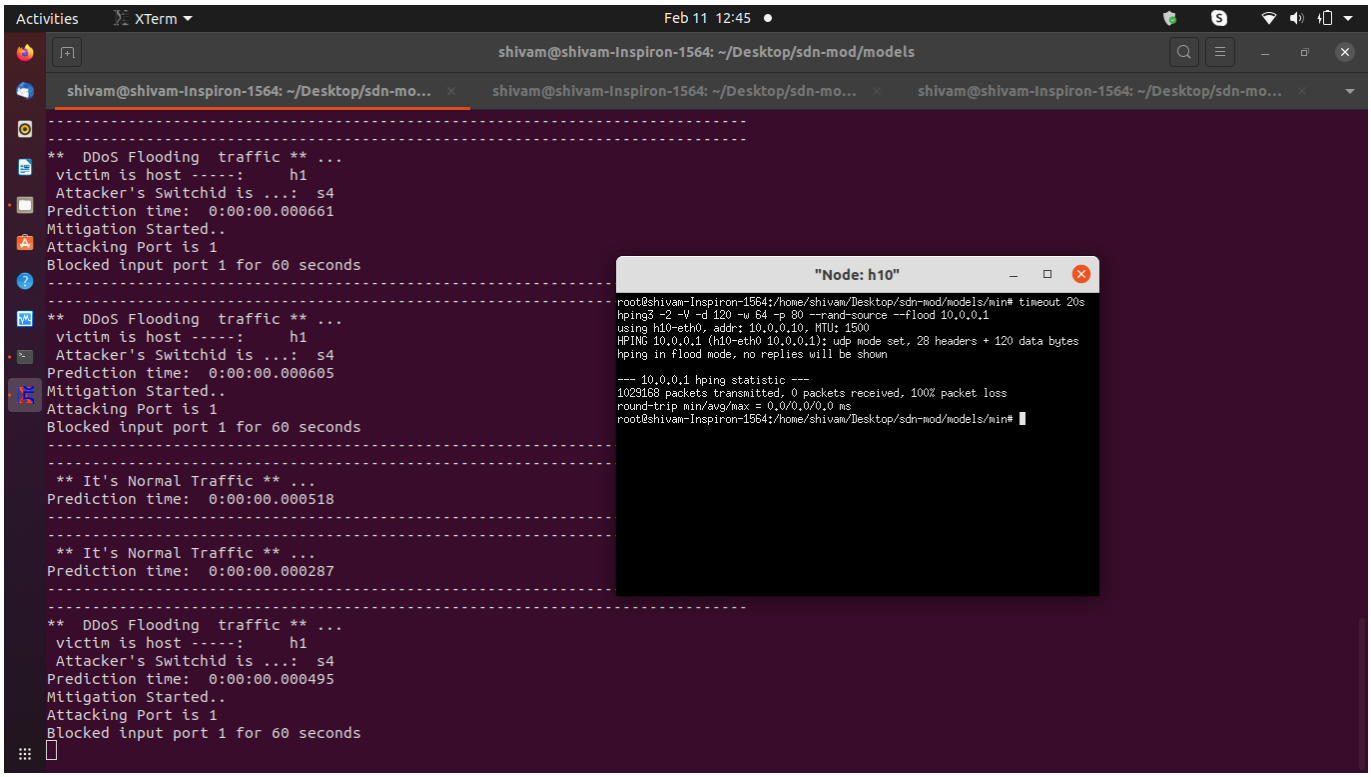

Fig. 9 Attack-Traffic Detection

Fig.9 depicts the sample attack traffic detection procedure; attack traffic was then launched from h10 to h1. When attack traffic is discovered, the detection program identifies the attacking switch and its input port. It then blocks the port for 60 seconds.

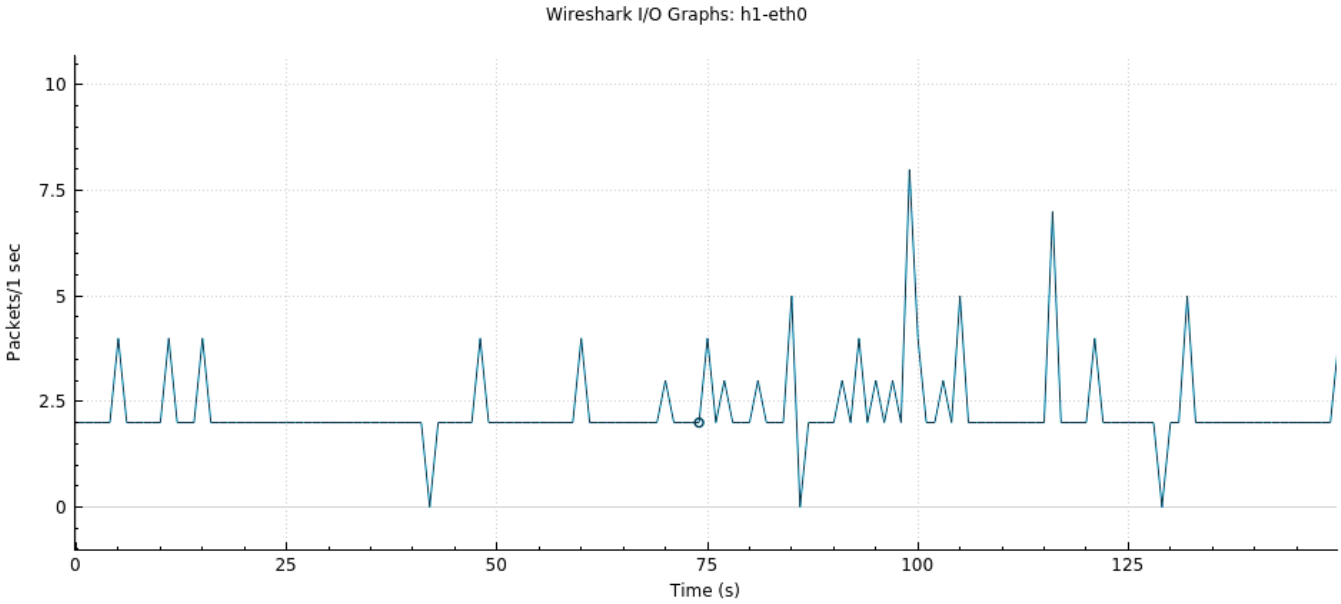

Fig. 10 Normal-traffic rate

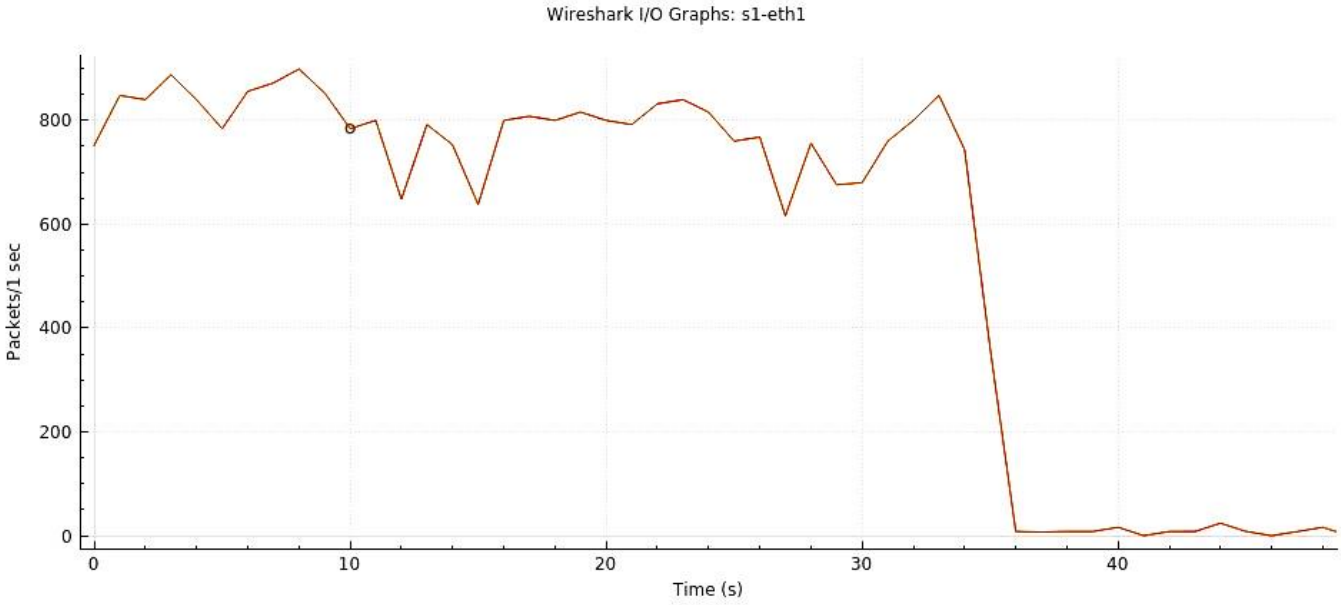

Fig. 11 Attack-traffic rate

The graph shown in Fig. 10 is an example of a real-time graph that depicts the typical flow of traffic as recorded by the wireshark analyzer.

On the other hand, Fig. 11 demonstrates that the amount of traffic increased at a far faster pace after the flooding attack was initiated by the attacking host and directed at the host (h1).

To compare our findings with those of similar studies, we created a comparison table. The results, as well as the ones reported in the literature, are summarized in Table 3.

The results reported in Table 3 suggest that the proposed machine learning model outperforms the findings presented in [11, 35, 36, 41, 43].

It is worth noting that similar research in the literature employed a wide range of datasets and models. As a consequence, substantiating the comparative findings is challenging.

The results reveal that by employing machine learning approaches, the SDN framework was successful in detecting DDoS assaults.

Table 3. Comparison of Proposed System with State-of-art

Ref	Dataset	Accuracy	Precision	Recall	F1-Score	Algorithm
[42]	MAWI Lab	93.79	NA	92.80	NA	Entropy
[34]	CCIDS2017	99.45	99.57	99.64	99.61	Ensemble CNN
[44]	Their own dataset	95.38	95.05	NA	NA	SVM
[43]	Their own dataset	85.49	NA	NA	NA	SVM-SOM
[23]	Their own dataset	99.99	100	99.98	99.99	AbaBoost
[11]	Their own dataset	98.85	99.03	98.74	NA	K-Means+KNN
[35]	NSLKDD	99.92	100	100	100	XGBoost
[35]	CICDDOS2019	99.9	100	100	100	XGBoost
[35]	CAIDA	99.7	100	100	100	XGBoost
[36]	Their own dataset	98.3	97.72	97.73	97.70	KNN
[37]	NSLKDD	98.78	98.7	96.1	93.3	J48
Proposed	Their own dataset	99.9	100	100	100	DT and XGB

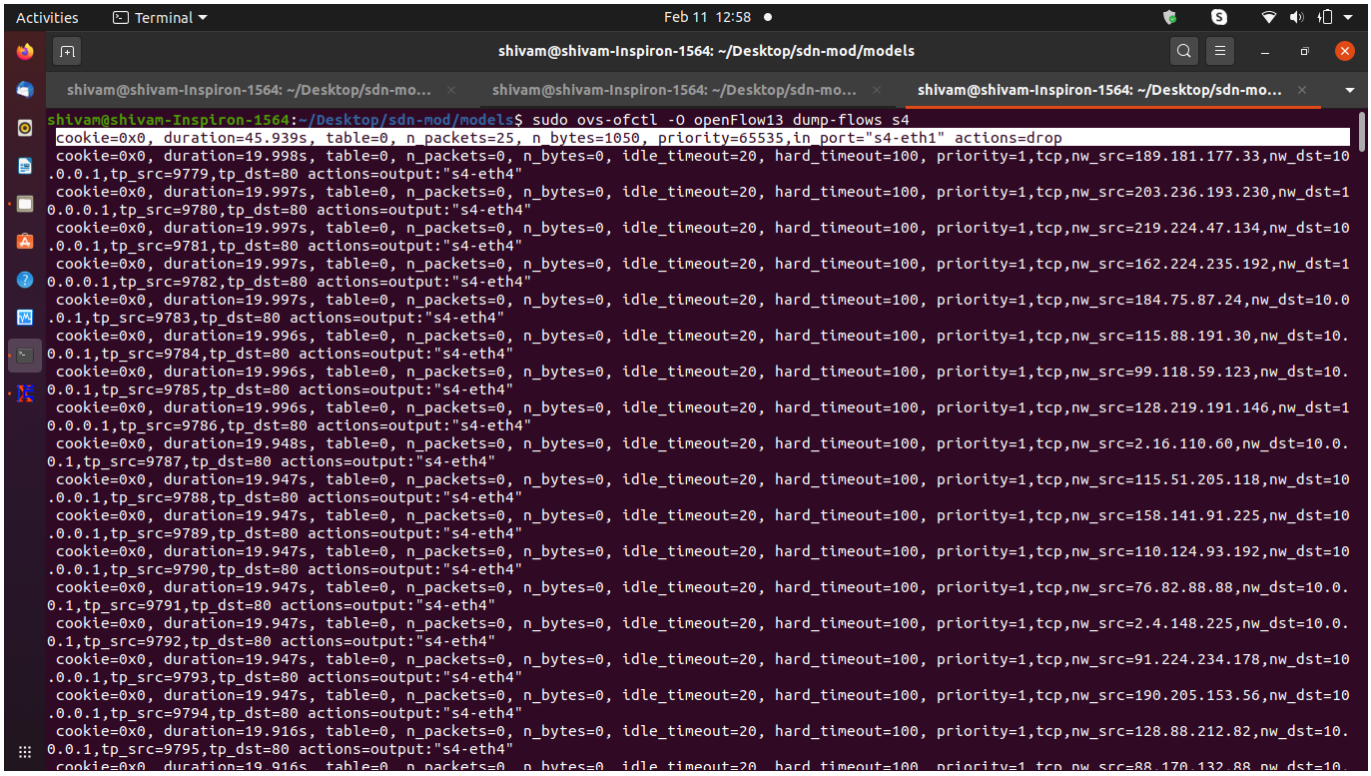

Fig. 12 Flow-Table Update Status

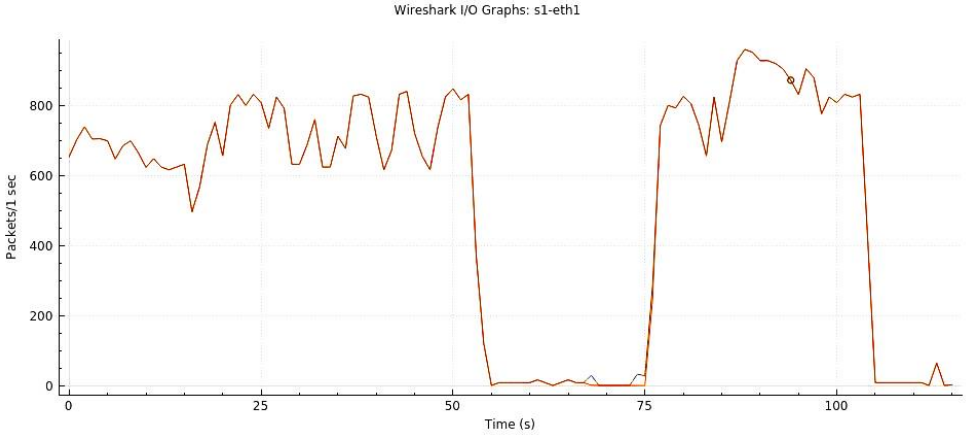

Fig. 13 Mitigation Traffic flow

5.4. Implementation of Mitigation Framework

The detection phase serves as the foundation for the mitigation step. Suppose the detection phase identifies potential attack traffic. In that case, a REST message is delivered to the controller and instructed to temporarily disable the DDoS port on the OpenFlow switch for a predetermined amount of time. If the anticipated traffic volume is within permissible limits, the controller does not get a command, and the computer's traffic flow is not disrupted. Following the discovery of DDoS flooding traffic, the status of the sample flow table update is shown in Fig. 12.

A dramatic decrease in traffic can be seen occurring in Fig. 13 immediately after the flow table is updated and the attacking port is blocked.

6. Conclusion and Future work

SDN has attracted much interest because of its many advantages; however, security is a major concern, and DDoS assaults are a formidable challenge for SDN. In this work, an XGBoost-Based Machine Learning (ML) model was created to detect and mitigate DDoS attacks on SDN-enabled systems. This was due to the combination of the strengths of XGBoost and its robust handling of large datasets, which made it a suitable choice for DDoS detection and mitigation.

Furthermore, the XGBoost algorithm has demonstrated effectiveness in binary classification tasks. This was critical in this case because the goal was to classify traffic as either normal or malicious. The procedure consists of traffic generation, collection, and classification, as well as detection and mitigation. According to the findings of the preceding investigation, the XGBoost model does better than the other machine learning models with regard to Accuracy, Recall, Precision, AUC-ROC, training time, and testing time. In the future, we hope to expand upon this work by building a deep learning-based detection system that can swiftly and accurately detect a wide range of DDOS attacks, implement mitigation strategies, and analyze performance across a range of multi-controllers and different SDN switch configurations.

References

[1] T Arvind, and Dr.K.Radhika, "Machine Learning Methods for Distributed DoS Attacks: Traffic Generation, Collection and Classification in an SDN Environment," *International Journal of Application or Innovation in Engineering & Management*, vol. 11, no. 8, pp. 1-8, 2022. *Crossref*, https://doi.org/10.2648/IJAIEM.1762.3462

[2] Kumar D, and Mrs. C. Veni, "IoE Security through Multi-Agent SDN," *International Journal of Computer Trends and Technology*, vol. 69, no. 12, pp. 5-9, 2021. *Crossref*, https://doi.org/10.14445/22312803/IJCTT-V69I12P102

[3] Dr.S.Kannan, and Mr.T.Pushparaj, "Creation of Testbed Security using Cyber-Attacks," *SSRG International Journal of Computer Science and Engineering*, vol. 4, no. 11, pp. 4-14, 2017. *Crossref*, https://doi.org/10.14445/23488387/IJCSE-V4I11P102

[4] K. Giotis et al., "Combining OpenFlow and sFlow for an Effective and Scalable Anomaly Detection and Mitigation Mechanism on SDN Environments," *Computer Networks*, vol. 62, pp. 122–136, 2014. *Crossref*, https://doi.org/10.1016/j.bjp.2013.10.014

[5] Seyed Mohammad Mousavi, and Marc St-Hilaire, "Early Detection of Ddos Attacks Against SDN Controllers," *Proceedings of the International Conference on Computing, Networking and Communications*, pp. 77–81, 2015. *Crossref*, https://doi.org/10.1109/ICCNC.2015.7069319

[6] Sangeetha M.V, and Bhavithra J, "Applying Packet Score Technique in SDN for DDoS Attack Detection," *SSRG International Journal of Computer Science and Engineering*, vol. 5, no. 6, pp. 20-24, 2018. *Crossref*, https://doi.org/10.14445/23488387/IJCSE-V5I6P104

[7] Fang-Yie Leu, and I-Long Lin, "A DoS/DDoS Attack Detection System Using Chi-Square Statistic Approach," *Systemics, Cybernetics and Informatics*, vol. 8, no. 2, 2010.

[8] Beny Nugraha, and Rathan Narasimha Murthy, "Deep Learning-based Slow DDoS Attack Detection in SDN-based Networks," *IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN*, pp. 51–56, 2020. *Crossref*, https://doi.org/10.1109/NFV-SDN50289.2020.9289894

[9] Anupama Mishra, Neena Gupta, and B. B. Gupta, "Defense Mechanisms against DDos Attack Based on Entropy in SDN-Cloud Using Pox Controller," *Telecommunication Systems*, vol. 77, no. 1, pp. 47–62, 2021. *Crossref*, https://doi.org/10.1007/s11235-020-00747-w

[10] Nisha Ahuja et al., "Automated DDoS Attack Detection in Software Defined Networking," *Journal of Network and Computer Applications*, vol. 187, p. 103108, 2021. *Crossref*, https://doi.org/10.1016/j.jnca.2021.103108

[11] Liang Tan et al., "A New Framework for DDos Attack Detection and Defense in SDN Environment," *IEEE Access*, vol. 8, pp. 161908–161919, 2020. *Crossref*, https://doi.org/10.1109/ACCESS.2020.3021435

[12] Trung V. Phan, and Minho Park, "Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud," *IEEE Access*, vol. 7, pp. 18701–18714, 2019. *Crossref*, https://doi.org/10.1109/ACCESS.2019.2896783

[13] Afsaneh Banitalebi Dehkordi, Mohammad Reza Soltanaghaei, and Farsad Zamani Boroujeni, "The DDos Attacks Detection through Machine Learning and Statistical Methods in SDN," *Journal of Supercomputing*, vol. 77, no. 3, pp. 2383–2415, 2021. *Crossref*, https://doi.org/10.1007/s11227-020-03323-w

[14] Nisharani Meti, D G Narayan, and V. P. Baligar, "Detection of Distributed Denial of Service Attacks using Machine Learning Algorithms in Software Defined Networks," *2017 International Conference on Advances in Computing, Communications and Informatics*, pp.1366-1371, 2017. *Crossref*, https://doi.org/10.1109/ICACCI.2017.8126031

[15] Mahmoud Said Elsayed, Nhien-An Le-Khac, and Anca D. Jurcut, "InSDN: A novel SDN Intrusion Dataset," *IEEE Access*, vol. 8, pp. 165263-165284, 2020. *Crossref*, https://doi.org/10.1109/ACCESS.2020.3022633

[16] T Arvind, and Dr.K.Radhika, "An SDN Based DDoS Traffic Generation, Collection and Classification Using Machine Learning Techniques," *International Conference on Advanced Engineering Optimization Through Intelligent Techniques*, Sardar Vallabhbhai National Institute of Technology, 2022.

[17] Obaid Rahman, Mohammad Ali Gauhar Quraishi, and Chung-Horng Lung, "DDoS Attacks Detection and Mitigation in SDN using Machine Learning," *IEEE World Congress on Services*, pp. 184-189, 2019. *Crossref*, https://doi.org/10.1109/SERVICES.2019.00051

[18] Reneilson Santos et al., "Machine Learning Algorithms to Detect DDos Attacks in SDN," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 16, 2020. *Crossref*, https://doi.org/10.1002/cpe.5402

[19] Boyang Zhang, Tao Zhang, and Zhijian Yu, "DDoS Detection and Prevention Based on Artificial Intelligence Techniques," *3rd IEEE International Conference on Computer and Communications*, pp. 1276–1280, 2017. *Crossref*, https://doi.org/10.1109/CompComm.2017.8322748

[20] Shi Dong, and Mudar Sarem, "DDoS Attack Detection Method Based on Improved KNN with the Degree of DDoS Attack in Software Defined Networks," *IEEE Access*, vol. 8, pp.5039-48, 2020. *Crossref*, https://doi.org/10.1109/ACCESS.2019.2963077

[21] Rochak Swami, Mayank Dave, and Virender Ranga, "Detection and Analysis of TCP-SYN DDos Attack in Software-Defined Networking," *Wireless Personal Communications*, vol. 118, no. 4, pp. 2295–317, 2021. *Crossref*, https://doi.org/10.1007/s11277-021-08127-6

[22] Filippo Rebecchi et al., "DDoS protection with Stateful Software-Defined Networking," *International Journal of Network Management*, vol. 29, no. 1, p. e2042, 2019. *Crossref*, https://doi.org/10.1002/nem.2042 .

[23] Rochak Swami, Mayank Dave, and Virender Ranga, "Software-Defined Networking based DDoS Defense Mechanisms," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1-36, 2019. *Crossref*, https://doi.org/10.1145/3301614

[24] Jupyter Notebook. [Online]. Available: https://jupyter.org/install

[25] Arvind T, and Dr.K.Radhika, "Comparative Assessment of SDN Openflow Controllers under Mininet Emulation Environment," *International Journal of Emerging Trends & Technology in Computer Science*, vol. 11, no. 4, pp. 80-84, 2022.

[26] Trung V. Phan, and Minho Park, "Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud," *IEEE Access*, vol. 7, pp. 18701-18714, 2019. *Crossref*, https://doi.org/10.1109/ACCESS.2019.2896783

[27] Sukhveer Kaur et al., "A Comprehensive Survey of DDoS Defense Solutions in SDN: Taxonomy, Research Challenges, and Future Directions," *Computers & Security*, vol. 110, p. 102423, 2021. *Crossref*, https://doi.org/10.1016/j.cose.2021.102423

[28] RYU SDN Framework Ryubook 1.0 Documentation. [Online]. Available: https://osrg.github.io/ryu-book/en/html

[29] Ryu Documentation. [Online]. Available: https://ryu.readthedocs.io/en/latest/getting_started.html

[30] Shi Dong, Khushnood Abbas, and Raj Jain, "A Survey on Distributed Denial of Service (Ddos) Attacks in SDN and Cloud Computing Environments," *IEEE Access*, vol. 7, pp. 80813–80828, 2019. *Crossref*, https://doi.org/10.1109/ACCESS.2019.2922196

[31] Introduction to Mininet, GitHub. [Online]. Available: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet

[32] Saurav Nanda et al., "Predicting Network Attack Patterns in SDN using Machine Learning Approach," *IEEE Conference on Network Function Virtualization and Software Defined Networks*, pp. 167-172, 2016. *Crossref*, https://doi.org/10.1109/NFV-SDN.2016.7919493

[33] Matheus P. Novaes et al., "Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment," *IEEE Access*, vol. 8, pp. 83765–83781, 2020. *Crossref*, https://doi.org/10.1109/ACCESS.2020.2992044

[34] Zohaib Hassan et al., "Detection of Distributed Denial of Service Attacks Using Snort Rules in Cloud Computing & Remote Control Systems," *IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control, IEEE*, pp. 283-288, 2018. *Crossref*, https://doi.org/10.1109/MSNMC.2018.8576287

[35] Hassan A. Alamri, and Vijey Thayananthan, "Bandwidth Control Mechanism and Extreme Gradient Boosting Algorithm for Protecting Software-Defined Networks against DDos Attacks," *IEEE Access*, vol. 8, pp. 194269–194288, 2020. *Crossref*, https://doi.org/10.1109/ACCESS.2020.3033942

[36] Huseyin Polat, Onur Polat, and Aydin Cetin, "Detecting DDoS Attacks in Software-Defined Networks through Feature Selection Methods and Machine Learning Models," *Sustainability*, vol. 12, no. 3, 2020. *Crossref*, https://doi.org/10.3390/su12031035

[37] Adel Alshamrani et al., "A Defense System for Defeating Ddos Attacks in SDN Based Networks," *MobiWac 2017 - Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access*, pp. 83–92, 2017. *Crossref*, https://doi.org/10.1145/3132062.3132074

[38] Peng Xiao, "An Efficient DDos Detection with Bloom Filter in SDN," *2016 IEEE Trustcom/BigDataSE/ISPA, IEEE*, pp. 1-6, 2016. *Crossref*, https://doi.org/10.1109/TrustCom.2016.0038

[39] Ahmed AlEroud, and Izzat Alsmadi, "Identifying Cyber-Attacks on Software Defined Networks: An Inference-Based Intrusion Detection Approach," *Journal of Network and Computer Applications*, vol. 80, pp. 152-164, 2017. *Crossref*, https://doi.org/10.1016/j.jnca.2016.12.024

[40] Shahzeb Haider et al., "A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks," *IEEE Access*, vol. 8, pp. 53972–53983, 2020. *Crossref*, https://doi.org/10.1109/ACCESS.2020.2976908

[41] Danish Sattar, Ashraf Matrawy, and OlufemiAdeojo, "Adaptive Bubble Burst (ABB): Mitigating DDoS attacks in Software-Defined Networks," *2016 17th International Telecommunications Network Strategy and Planning Symposium*, pp. 50-55, 2016. *Crossref*, https://doi.org/10.1109/NETWKS.2016.7751152

[42] Kübra Kalkan et al., "JESS: Joint Entropy-based DDoS Defense Scheme in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2358–2372, 2018. *Crossref*, https://doi.org/10.1109/JSAC.2018.2869997

[43] V Deepa , K. Muthamil Sudar, and P Deepalakshmi, "Detection of DDoS Attack on SDN Control Plane using Hybrid Machine Learning Techniques," *Proceedings of the International Conference on Smart Systems and Inventive Technology*, pp. 299-303, 2018. *Crossref*, https://doi.org/10.1109/ICSSIT.2018.8748836

[44] Aye Thandar Kyaw, May Zin Oo, and Chit Su Khin, "Machine-Learning Based DDOS Attack Classifier in Software Defined Network," *The 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pp. 431-434, 2020. *Crossref*, https://doi.org/10.1109/ECTI-CON49241.2020.9158230

[45] Tsung-Han Lee, Lin-Huang Chang, and Chao-Wei Syu, "Deep Learning Enabled Intrusion Detection and Prevention System over SDN Networks," *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1-6, 2020. *Crossref*, https://doi.org/10.1109/iccworkshops49005.2020.9145085

[46] Dragos Comaneci, and Ciprian Dobre, "Securing Networks Using SDN and Machine Learning," *IEEE International Conference on Computational Science and Engineering, IEEE*, pp. 194–200, 2018. *Crossref*, https://doi.org/10.1109/CSE.2018.00034

[47] Song Wang et al., "Detecting Flooding DDos Attacks in Software Defined Networks Using Supervised Learning Techniques," *Engineering Science and Technology, An International Journal*, vol. 35, p. 101176, 2022. *Crossref*, https://doi.org/10.1016/j.jestch.2022.101176

[48] Rui Wang, Zhiping Jia, and Lei Ju, "An Entropy-Based Distributed DDos Detection Mechanism in Software-Defined Networking," IEEE Trustcom/BigDataSE/ISPA, Helsinki, pp. 310–317, 2015. *Crossref*, https://doi.org/10.1109/Trustcom.2015.389