International Journal of Advanced Technology and Engineering Exploration, Vol 8(76)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/IJATEE.2021.874021

Research Article

Detection and mitigation of botnet based DDoS attacks using catboost machine learning algorithm in SDN environment

Sanjeetha R*, Anant Raj, Kolli Saivenu, Mumtaz Irteqa Ahmed, Sathvik B and Anita Kanavalli
Department of Computer Science and Engineering at MS Ramaiah Institute of Technology, India

Received: 08-February-2021; Revised: 18-March-2021; Accepted: 21-March-2021

Abstract
Software-defined networking (SDN) is an emerging new technology in the field of networks that facilitates comprehensive network programmability, which makes them prone to network attacks. One of the primitive yet highly effective network attacks is the Distributed Denial-of-Service (DDoS). DDoS attacks are launched from the compromised hosts called botnets acquired by the attacker host called the botmaster, all being connected to switches present in the same environment. Despite the large number of traditional mitigation solutions that exist today, DDoS attacks continue to grow severely. Numerous solutions have been proposed to counter these attacks and prevent service disruptions which have cost many companies a fortune. An extensive literature survey of existing solutions to these security challenges in an SDN environment, that employed machine learning techniques like XGBoost, Support Vector Machine (SVM), etc., has addressed the detection of DDoS attacks. But still showed the scope of improvement in detection speeds which could significantly reduce the service unavailability time from a server i.e., the victim of the DDoS attack. Thus, this paper addresses these requirements to build an optimal, reliable, and quick DDoS detection and mitigation application. This application leverages the controller's functionalities, continuously monitors the network traffic at a particular host interface (potential victim) to detect abnormal traffic. When the traffic is identified as a potential DDoS attack, its mitigation is initiated. The DDoS attack traffic is mitigated by deploying flow rules onto the switches such that it blocks the attack traffic from entering the network. The application uses CatBoost classifier, the boosting algorithm which has very less prediction time and is comparatively 8× faster than XGBoost, because of its symmetric tree structure. It is tested to be proven reliable and efficient in detecting botnet-based DDoS attacks on the SDN environment with an accuracy of 98% and far less training time. Thus, proving that the proposed solution employing the state-of-the-art machine learning model can be more effective in quickly detecting and mitigating a DDoS attack.

Keywords
SDN, Botnet, DDoS, Machine learning, Catboost.

1.Introduction

1.1Background

Software-Defined Networks is an emerging new technology in the field of networks that had a lot of impact in fields such as cloud computing and data centers. In traditional networks, the data plane, which is responsible for the forwarding of data, and the control plane, which decides the path of the packet are tightly coupled. But in software-defined networks, the control plane and data plane are separated. A special device called the controller acts as the control plane. The switches act as the data plane which forwards the packets based on the flow rules defined by the controller.

The controller is connected to all the switches and the controller communicates with the switches using a secured protocol called OpenFlow. Such a system encourages modularity, freedom to choose the software and the hardware, and is quite robust when deployed as a network.

The introduction of SDN brought in certain advantages that were absent in traditional networks, but such networks are still susceptible to DDoS attacks which can disrupt all the services in the network. This requires the SDN to have an efficient, quick and accurate detection and mitigation mechanism for such attacks. The use of XGBoost is proposed for the purpose of DDoS attack detection and has been observed to outperform traditional algorithms such as Support Vector Machine (SVM) and random forest in terms of accuracy and speed [1].

*Author for correspondence

The detection process can be improved further by trying out newer algorithms like CatBoost which are supposed to perform better than the ones that are proposed and mentioned.

Research has been done with the use of a third-party application on top of the existing network topology for mitigation of the attack by determining the attacker address with the help of the traffic information collected while monitoring the network flow and a Firewall is used to drop the attack packets from identified source [2]. This method suggests a way to stop the DDoS attack by analyzing network traffic and provides motivation for development of a similar application on top of the SDN that handles the mitigation part.

Today, SDN has attracted a lot of industrial and academic interest as a digital technology that promotes network management and offers new ways to dynamically control and execute networks. DDoS attacks are more prevalent in traditional networks. One way of performing a DDoS attack is using botnets. An intruder can create botnets by installing malware into the hosts and later use them to perform DDoS attacks on a server. Such attacks can also be used on a server present in SDN environment. While there are many standard mitigation techniques, DDoS attacks are still widespread today. Many ideas have been suggested to combat these attacks and to discourage service disturbances that have cost a lot to many businesses. A comprehensive literature study on current solutions in an SDN environment to such security problems, using machine-learning techniques like XGBoost, SVMs, and so on, showed that such techniques tackled DDoS-attacks well, but there is still scope for improvement for attack identification, which could greatly reduce the service disruption time for the server.

1.2Machine learning models

The machine learning models are of two types. One is the supervised learning model and the other is the unsupervised learning model. A supervised learning model has predefined labels or classes in the dataset that are to be predicted, whereas the unsupervised learning model finds hidden structure in a given dataset without the help of any labels.

Any of the above-mentioned models need a dataset for training. A dataset can differ based on our objectives and the type of learning model that we are making. The dataset is pre-processed and the most important attributes affecting the model are derived. The model is trained on a certain ratio of the dataset and the rest is used for testing purposes. Based on test set results, we determine the accuracy of the model. The model can be used in different files by converting it into the form of a package. Usually, the model is converted to a pickle file and is used in the application. Some of the supervised learning models used throughout the experimentation are mentioned below.

Logistic regression is one of the simplest models of the classification problem. In this model, the dataset is used to train the various parameters/coefficients of the equation of the model, and the output of the model are passed as input to the sigmoid function to obtain a value between 0 and 1. The obtained value gives the probability of whether the hypothesis of the model is true.

The decision tree algorithm is a supervised learning algorithm that can be used for both classification and regression tasks. In this model, a treelike graph is created in which each node represents the attribute, each edge represents a value in the parent attribute node, it is connected to and the leaf nodes represent the classes of the classification problem. Mathematical techniques such as the Gini index, information gain is used to find the most important attributes of the classification problem, and the tree is built accordingly from top to bottom. Each new tuple starts traversing from the root and follows the path based on its attribute values till it reaches the leaf which gives the class prediction.

The Gaussian Naive Bayes algorithm is another supervised learning algorithm that works on the principle of the Bayes theorem of probability to predict the unknown class. It assumes that every attribute used for prediction is independent of each other and that each attribute makes an equal contribution to the prediction of the class. Since the dataset used has real value attributes, the Gaussian Naive Bayes are used for prediction. In this method, the posterior probability of each hypothesis/class is calculated using the prior probability and the hypothesis/class which has the highest priority is taken as the prediction.

XGBoost is another supervised algorithm that uses gradient boosted decision trees. It is one of the most powerful machine learning algorithms out there. It uses the principle of gradient boosting to get better results over convention decision trees. Boosting is an ensemble technique employed to increase the model

accuracy by retraining the model on incorrectly predicted tuples in the training phase to get the correct outcome for the tuple. Gradient boosting is a new approach where new models are created that predict the errors of the previous models and the cumulative results of all the models are taken together to give the final prediction. It is named gradient boosting as it makes use of a gradient descent algorithm to minimize the loss in the model during training.

K-Nearest Neighbors is a supervised learning algorithm in the "lazy learners' category". Each time a new tuple is fed to the model, the model calculates the distance between the tuple and all the other tuples in the dataset and takes the K-nearest tuples and the class that occurs the highest number of times in the k-tuples is assigned as the prediction to the new tuple. For this algorithm, K is an important parameter that must be determined for the algorithm for high accuracy.

CatBoost, which is observed to be superior when compared to the other models in this experiment, is a new open-sourced supervised learning algorithm developed by Yandex in 2017. It utilizes gradient boosting on decision trees. It has a faster training time when compared to other gradient boosting algorithms. Another unique feature of this algorithm is that it will automatically handle categorical features by converting them into numerical features.
Thus, this paper proposes an optimal and reliable DDoS detection and mitigation technique to prevent DDoS attacks. The technique design and development is done after extensively studying the existing areas of research in a similar field of interest. Comparisons of the advantages and disadvantages of the existing or proposed solutions using multiple Machine Learning models for detecting the DDoS attack in an SDN environment paved the way for the development of the detection and mitigation module that tries to overcome these disadvantages and provide for a more efficient and reliable solution. The technique involves simulating a Botnet-based DDoS attack in an SDN environment, constructed using Mininet. The controller used for the SDN network is the RYU controller. One host is considered as the Botmaster which then takes control of some of the other hosts through SSH connections, constituting the botnet. These infected hosts then initiate a UDP-Flood DDoS attack on the victim server.

A machine learning algorithm (CatBoost) is used to detect such attacks in real-time. A Software Defined Network Application written in python that uses the Catboost model for the detection of a DDoS attack is deployed as an external module that interacts with the controller using REST APIs querying for network traffic data. Once a DDoS attack is detected in the network, the hosts responsible for the attack are identified and then the mitigation of the attack is performed by installing flow rules into switches. These flow rules deployed, block all the incoming traffic from the infected hosts, and thus the DDoS attack is mitigated.

The main contributions of this paper can be summarized as follows:

- Development of a novel and faster application, built on top of the controller for DDoS detection.
- Using state-of-the-art machine learning techniques for increased detection efficiency.
- Implementation of a flow rule generation scheme to mitigate the effects of the DDoS attack by disrupting the connection between the botnet and the victim within the SDN.
- Evaluation of the performance of the proposed solution by testing in a simulated environment by inducing a DDoS attack.
- Visual Representation of the performance of the network using suitable graphs.

2.Literature survey

Chen et al., have used XGBoost classifier for classifying whether the attack is a DDoS attack or not. XGBoost is an ensemble of decision trees that uses boosting principles to improve performance. In comparison, it was noted that while Support Vector Machine, Random Forests, and Gradient Boosted Decision trees gave an accuracy of 97.19%, 96.33%, and 97.69% respectively, XGBoost topped with 98.53% accuracy. The running time of XGBoost is only higher than Random Forest with a time interval of 11 seconds. Overall, from all the models tested, it was inferred that XGBoost gave the best results [1].

Another method uses the traffic analyzer to collect flow information in a synchronized manner and incoming packets are compared with expected packets. The top command is used to monitor network traffic and forward results to another file and analyses are done to check for an attack. To reduce false alarms, the next flow is monitored to check whether packets are still being sent by a particular address. If yes, then the address is forwarded to the firewall so that packets from the attacker can be dropped otherwise the client will be considered a normal client. A firewall is placed in the controller

(POX controller in this case). The performance overhead was observed to be low due to less CPU usage as the firewall program was used only in the controller and not in the clients [2].

Reflective DDoS attacks are those in which the attacker sends a request packet to an exploitable proxy server with a spoofed IP address (which is the IP address of the victim to be attacked). This proxy server sends the response messages to the victim, which results in exhaustion of its resources. To detect and mitigate such an attack, incoming packets are classified as legitimate or illegitimate using NAT (Network address translator). A differentiator separates requests from the responses as Reflective attacks are possible only by the responses. After detecting an attack, the mitigation system comes into action which updates flow rules to block attacking traffic [3].

A way to organize a DDoS attack is the Slow HTTP DDoS attack, in which incomplete HTTP GET messages are sent to the target server. The server maintains these connections expecting the sender to complete the message. If a lot of such messages are sent by different hosts (through a botnet), it can consume the resources of the target, and thus legitimate clients cannot access the server. The mitigation system proposed uses a threshold for the number of open connections, and if at any given time, the number of open connections exceeds the threshold, the controller with the help of SHDA (Slow HTTP DDoS Defense Application) performs timeout-based attack detection and isolates the attackers [4].

Deepa et al. [5] have proposed a 2-model concept for SDN attacks. One model is a SVM model followed by a Self-Organizing map model; SVM is used to detect the DDoS attacks that it has learned from the dataset whereas Self Organizing Map model can be used to detect new types of attacks. While SVM and Self Organizing Maps independently gave an accuracy of 82.31% and 93.243%, respectively, a combination of the 2 gave an accuracy of 98.12%.

Lawal and Nuray [6], have proposed a method in which threshold value T is used and once the traffic in the network crosses the threshold T, then the sFlow management system generates traffic rules for handling such high traffic and sends it to the controller. The sFlow is a real-time traffic sampling technology that is used for monitoring traffic in the network. The controller then sends OpenFlow rules to the switches instructing them to drop the malicious packets, thus reducing the effect of the attack. The method was tested in a Mininet network topology using a floodlight controller. An ICMP flood attack was used for simulating a DDoS attack and graphs were generated to analyze the traffic and flow of packets.

Wijesinghe et al. [7] have proposed a method for the detection of a range of botnets in an SDN environment. The traffic flow among hosts is recorded according to the IPFIX template and that information is used to detect specific features of the bots with the help of machine learning techniques. If any of the hosts are found to be infected, then the infected hosts are removed from the rest of the network environment. It was observed that different techniques were helpful to detect different ranges of botnets and no specific technique was found to be effective for detecting all the families.

Dao et al. [8], present an approach to combat flooding attacks against the controller in which spoofed requests are sent continuously, causing a burden on the controller-switch channel and overloading the flow table in the switch leading to a downgrade of the quality and stability of the network. First, the SDN environment was simulated with regular legitimate traffic, and also with a DDoS attack against the controller, and analysis was performed on the traffic. Let 'n' be the minimum number of packets sent in a connection and 'k' be the total number of connections held by a legitimate user. An attacker would have more than 'k' connections and transmit less than 'n' packets per connection. The number of unique packets sent from an IP address is stored by a counter 'c'. For a new packet, the system increments the value of 'c' of the corresponding IP address and if it is greater than 'k', the traffic from that IP address is analyzed with an average number of packet counter 's'. If 's' is found to be less than 'n', it is assumed to be an attacking packet, and further, all packets originating from that IP address are blocked by a suitable flow rule. Thus, the controller is safe against DDoS attacks.

The existing methods used for attack detection with KNN had high accuracy, but required more technology to determine the threshold and weight distribution. Dong and Sarem [9] have proposed a DDoS Detection Algorithm based on the Degree of Attack (DDADA) and DDoS Detection Algorithm based on Machine Learning (DAMDL) to improve the current condition. The attack detection method

based on an improved KNN with Degree of DDoS Attack data plane is a combination of forwarding elements used to forward traffic flows based on instructions from the control plane. They have also classified attacks into four major groups as HTTP, ICMP, UDP and SIP flood attacks. Four features are analyzed when the SDN controller is attacked, that is flow length, flow duration, flow size, and flow ratio. For detection, the DDADA algorithm is proposed and DDAML is introduced to further improve the efficiency. The performance is evaluated on the accuracy of detection, ROC and AUC metrics. The proposed solution is seen to perform more effectively than the traditional algorithms like NB algorithm, SVM algorithm, and others based on the obtained TPR, FPR, precision, F-measure and recall values.

Yadav and Selvakumar [10], have used the Logistic regression algorithm to detect Application layer DDoS attacks. The dataset to train the algorithm was generated by simulating normal and attacking traffic on the nitt.edu website. The features obtained through simulation were further processed to construct more features for better analysis. The principal component analysis was further done to find the most important features. On this modified data, a Logistic Regression algorithm was applied and an Accuracy of 98.6% with a False Positive Rate of 1.41% was obtained.

Traditional detection systems employ Packet level analysis and payload examination as the dominant methods for identifying malicious network traffic. The system examines all the incoming packs for any suspicious activity. The traditional methods are ineffective in detecting intrusions as most of the new DDoS attacks mimic legitimate web service traffic. Fouladi et al. [11] propose a stand-alone frequency analysis method for DDoS attack detection in which the traffic flow level of the network is analyzed. The DDoS attack is separated from normal traffic using coefficients of Discrete Fourier transform (DFT) and discrete wavelet transform (DWT). The accuracy of the detection is increased by using Wavelet transform. This is because it provides higher resolution information about the frequency domain. The separation between attack and normal traffic is done using a Naive Bayes classifier that has two frequency-based methods of DFT and DWT and results are compared with a simple thresholding classifier. The dataset has 1936 and 2649 samples of normal and attack traffic respectively. Three different feature sets including DFT, DWT, and DFT+DWT (combined feature) are provided to the classifier. In comparison to other features, the combined feature improves accuracy with the lowest false positive and false negative rates.

Lakshminarasimman et al. [12] have compared two variants of decision tree classifiers which are J48 algorithm and Random forests for detecting the DDoS attack. J48 works in the same way as ID3 but uses the concept of information entropy to make the decision tree. Random forest algorithm consists of an ensemble of decision trees in which each tree gives a prediction and the prediction that occurs the most is chosen as the final prediction. The two models were trained on the KDD 1999 dataset. 10-fold cross-validation was applied in the experiment for accurate evaluation. It was found that the J48 algorithm with an accuracy of 99.9415% gave better results than the random forest algorithm that gave 96.94%.

This paper detects the attack traffic through the central SDN controller. In this study, a SVM is used in conjunction with a kernel principal component analysis (KPCA) and a genetic algorithm (GA) to improve detection accuracy. SVM techniques are the prime classifier for malicious traffic prediction. An effective way of protecting SDN was proposed and analyzed by three different variants of SVM. SVM with KPCA and GA are combined with the proposed approach of detection. KPCA is executed for feature extraction, and the SVM classification is used for the classification of attacks. Also, the feature differences are reduced by an enhanced Radial Basis Kernel Function (N-RBF). Genetic algorithms are also used to optimize various classifier parameters. The experimental results demonstrate that the proposed model is better classified with better generalization in comparison with the single SVM. Also, the model proposed can be integrated into the controller to specify security rules to avoid potential attacks by attackers [13].

This paper emphasizes that although extensive studies have been carried out on Denial of Service (DoS) attacks and mitigation of the DDoS attacks, such attacks remain difficult to be detected. This paper presents a flexible, modular architecture, which allows LR-DDoS attacks in SDN settings to be identified and mitigated. In particular, the authors use six ML model (i.e., J48, Random Tree, REP Tree, Random Forest, Random Perceptron (MLP), and SVM) in their architecture, to train the intrusion detection system (IDS), and to evaluate the performance using the CIC-DoS datasets. The evaluation results show that, despite the difficulty in identifying LR-DoS attacks, the approach achieves a

detection rate of 95 percent. For the simulated environment to be as close to real-world production networks, the authors also point out that, with their deployment, they use Mininet virtual machine operating system with an open network operating system, ONOS. The intrusion detection system mitigates all previous IDS attacks with their testing topology. This shows how useful the proposed architecture is for identifying and mitigating LR-DDoS attacks [14].

Many Internet protocols have a range of flaws that attackers can take advantage of to launch a series of attacks. DNS, one of the Internet's most important elements, is one of these protocols. Because of the User Datagram Protocol (UDP) exchanges in this protocol, it is primarily vulnerable to DDoS attacks. These attacks are difficult to counter because attackers spoof the victim's IP address and flood it with valid DNS responses from legitimate DNS servers. The authors of this paper suggest WisdomSDN, an inexpensive and scalable solution for effectively mitigating DNS amplification attacks in SDN. WisdomSDN detects and mitigates illegal DNS requests and responses. WisdomSDN is made up of two parts: (1) a novel proactive and status-based (PAS) program for one-to-one computer-generated mapping of DNS requests with DNS responses; (2) a machine learning DDoS detection module for detecting illegal DNS requests only in realtime. This module consists of (a) Flow Statistics Collection Scheme (FSC) for effective and scalable collection of flow features through the sFlow protocol; (b) To assess the randomness of network traffic, an entropy estimation scheme (ECS) is used.; and (c) BNF is a Bayes Network-based Filtering system that uses entropy values to distinguish illegitimate DNS requests.; and (3) DNS Mitigation (DM) is a DNS mitigation scheme that essentially mitigates illegitimate DNS demands. The experimental findings indicate that WisdomSDN can effectively detect/mitigate DNS amplification attacks rapidly with a high detection rate, low false-positive rate, and less overhead as compared to the state-of-the-art, making it a promising solution to mitigate DNS amplification attacks in an SDN setting [15].

The lack of trust evaluation and management mechanism between the OpenFlow switches is a major concern in SDNs especially when it comes to dealing with DDoS attacks. Hence to tackle this issue, a trust evaluation, and management model is proposed which is called the Intelligent Trust Model (ITM). The ITM consists of a network monitoring module and a trust evaluation module. The network monitoring module monitors network parameters like packet loss rate and time delay. Based on the parameters, trust evaluation is performed by the network intelligent trust module. The trust value is used to measure the degree of trust among OpenFlow switches. Based on real-time updation of trust values, the hybrid-DDoS attacks are detected quickly with Extreme Learning Machine (ELM). The model was tested on multiple types of DDoS attack environments to ensure the authenticity and effectiveness of obtained results [16].

To tackle the issue of DDoS attacks in the SDN-based cloud environment, a hybrid machine learning model has been proposed which is based on a support vector machine and self-organizing map algorithm which helps in enhancing the classification of the network traffic. A statistic sender sends a request to the OpenFlow switch which returns response data and that data is processed in the Raw Data Processing module. Then the processed data is sent to an appropriate and trained SVM classifier. Another IP filtering scheme is introduced that is based on enhanced history which improves the rate of detection of DDoS attacks. It uses a predefined set of parameters to distinguish between normal and attack source. The combination of the mentioned techniques provides a DDoS attack defender which is used in the SDN cloud-based environment. The proposed solution aims to deliver protection against attacks along with ensuring a better quality of service to cloud customers [17].

The vulnerability of SDNs due to the presence of a single point of failure by DDoS attacks is proposed to be tackled with the help of a defense and attack detection framework in the SDN environment. A periodic trigger is used for the detection of DDoS attacks and the detection cycle period plays an important role in the detection efficiency and the controller's performance. The detection trigger mechanism is deployed in the data plane to reduce the resource burden on the controller and minimize communication overhead between the controller and the switches. The classification-based detection method requires the selection of important flow features that affect the accuracy of the algorithm. These selections are made based on certain parameters such as average byte stream rate, stream duration, percentage of symmetric flows, and traffic surge analysis. Finally, a combined machine learning algorithm that consists of K-Means and KNN is used to detect DDoS attacks and maintain a proper balance

between the accuracy and efficiency of detection. Once the attack is detected, it is mitigated by the controller adding a flow entry in the switch to drop attack packets. The malicious flow entries are also removed to release occupied storage space [18].

Alamri and Thayananthan [19] have worked on a DDoS detection and mitigation system which comprises 2 components, a bandwidth control mechanism and a machine learning-based detection system that uses the XGBoost model. In the bandwidth control mechanism, 3 threshold profiles would be calculated to classify the flow based on time and byte rate. The three thresholds are low traffic threshold profile, medium traffic threshold profile, and heavy traffic threshold profile, which will be used based on the time of the day. Any time the network flow crosses the threshold chosen based on the time of the day, the bandwidth for that flow is reduced by half. The detection system consists of the XGBoost model that is used to classify whether a flow is normal or malicious. The system consists of 3 main phases, the monitoring phase where the network flow is compared with the threshold to see if it exceeds or not, the bandwidth control phase where the bandwidth is halved and a counter is maintained to calculate how many times the traffic has exceeded the threshold value, and finally, the detection and mitigation phase, where the flow statistics are sent to the XGBoost classifier to classify if the flow is normal or malicious, once the counter reaches a threshold. Various machine learning models such as Logistic Regression, Naive Bayes, random forest, etc were trained and tested but finally, the XGBoost model was chosen on account of very high accuracy. Wang et al, discuss a new type of DDoS attack called link flooding attack (LFA) that targets high traffic, vulnerable links in the network and suggest a novel software designed by them called LDADefender, to detect and mitigate them in an SDN environment. LFA has 2 crucial characteristics, firstly, it uses large scale slow-speed legitimate flows to initiate the attack, and secondly, due to its adaptive nature, it can change the target victim link in real-time thus making it difficult for traditional defense mechanisms to detect it. The software consists of 4 stages, namely, target link selection, where the potential victim links with high flow density are identified, link congestion monitoring, where a monitoring agent is deployed at each identified target link to be able to capture and send the flow statistics from the link to the controller, traffic rerouting, where the traffic at congested links are rerouted to decongest the link and to temporarily mitigate the DDoS attack and finally the malicious traffic blockings where the LFA bots are identified and cut off from the network using flow rules. The LDADefender was tested on CloudLab and it could successfully detect bots with an accuracy of over 90pc [20].

Jia et al. [21] have proposed a defense mechanism against IoT-based DDoS attacks called Flowguard, which comprises two modules, Flow Filter and Flow Handler. The Flow Handler outputs flow rules based on the network traffic and the Flow Filter filters the incoming packets based on these generated rules. The rules are generated using self-evolving machine learning/deep learning models. The dataset used is the CICDDoS2019 dataset, which contains various types of DDoS attacks, such as Flooding Attacks as well as Slow request/Response attacks. The network traffic is generated using edge devices (IoT Devices), and this data is processed and analyzed at edge servers. Two models are used, a Long Short-term Memory (LSTM) model to identify an attack and a Convolutional Neural Network (CNN) to classify the type of attack. The model developed performed at an accuracy of 98.9% in identifying new attacks, and 99.9% in classifying the exact type of attack. Thus, Deep Learning models have been used to effectively detect and mitigate IoT-based DDoS attacks.

3.Methodology of proposed solution

The proposed system consists of the SDN environment and detector and mitigator application of which a Machine Learning model is a part that monitors network traffic to determine when an attack takes place. The solution proposed consists of the following modules:

3.1Dataset acquisition and pre-processing

To train the Model, a vast dataset consisting of various types of network attacks is needed. This is obtained from the "DDoS attack network logs" - an open dataset from Kaggle. The dataset consists of over 2.1 million entries which are sufficient to train a model effectively. Several pre-processing techniques are applied to the dataset to obtain data in the required order for the Model. Feature extraction is also an important step so that only data that is necessary to detect an attack can be used to train the model.

3.2Training and Testing

The pre-processed dataset is split in the ratio of 80: 20 where 80% of the data is used for training and 20% is used for testing. The testing dataset is used for cross-validation, thereby helping in the analysis of

the model. This training dataset is trained on various models, such as Logistic regression, Naive Bayes classifier, CatBoost classifier, etc. to find the best performing model which took the least amount of training time.

3.3 Predictions and analysis

The model generated is then used for real-time predictions on the network traffic generated by the SDN environment and used in determining whether the traffic flow in the SDN network is attacking traffic or not. Performance-based on various criteria such as accuracy, precision, recall, time is analyzed using various visual methods such as line plots, bar graphs, etc.

3.4 Setting up the environment

Mininet is used to set up the required SDN environment for testing purposes. It creates a network of hosts, switches, controllers, and links.

3.5 Setting up SSH connection and traffic generation

The botmaster uses SSH to set up a connection with the botnets. SSH is a protocol that allows a secure connection between a client and a server over an unsecured network. The compromised hosts then generate traffic with the help of hping3 which is a command line-oriented TCP/IP packet assembler. It supports TCP, UDP, ICMP, and RAW-IP protocols.

3.6 Detection and mitigation

The DDoS attack is detected by the machine learning model. The flow stats are retrieved with the help of the rest APIs provided by the RYU controller. RYU is a python-based controller which provides northbound APIs to allow communication between the application and the controller to fetch flow statistics and add/modify flow rules. Mitigation is achieved by updating the flow rules of the identified victim and the botnets after an attack is detected.

Algorithm: Algorithm for detector-mitigator module
Input: Packet Rate, Byte Rate
Output: Mitigates DDoS attack
Initialization:
Load machine learning model
Setup API to make requests
while (true) do
Make a GET request to the Mil.") Northbound APIs to retrieve flow statistics.
Packet Rate <- Packet count total duration in second
Byte Rate <- Byte count total duration in second
src (- source mac address
dst <- destination mac address
ML model input = Packet Rate, Byte Rate
if (model detects DDoS attack) then
add new flow rules to block connection between src and dst by making POST request to RYU Northbound APIs
end if
end while

4.Implementation

4.1System configuration

The SDN topology was set-up and tested on a virtual emulator called mininet which was running on a system with Ubuntu 18.04 Operating System. The system had 12 GB of RAM, i5 7200U CPU which is a dual core processor and 2GB of dedicated NVIDIA 940MX GPU. For the purpose of the SDN controller, RYU was used which is a python-based controller.

4.2Data exploration and feature extraction

The dataset used consists of 28 attributes such as “Source address”, “Destination address”, “Node Name from”, “Node Name to”, etc. The target is a string that denotes the type of packet such as “Normal”, “UDP Flood”, “HTTP Flood”, “SIDDOS” or “Smurf”. The total number of entries in the dataset is 2,160,668. The dataset is examined to find any missing entries, and all such entries are discarded. The string values are in byte format, and thus have to be first decoded into “UTF-8” format. The target attribute is then converted into integer values as most Machine Learning algorithms need the target attribute to be a numerical value. A statistical description of the dataset, using PACKET RATE and BYTE RATE as the main parameters is shown in *Tables 1* and *Table 2*. It includes the values of the total number of data entries, mean, standard deviation, minimum value, quartiles, and maximum value.

Table 1 Statistical description of the dataset with respect to 'PKT_RATE'

Class	Count	Mean	std	min	25%	50%	75%	max
Normal	1935959	288.5207	91.05149	0.9779570	328.0642	328.2178	328.4318	658.0904
HTTP-Flood	4110	26.928764	16.124078	22.967447	23.041765	23.108231	23.175081	94.721200
UDP-Flood	201344	940.048809	221.670555	0.977957	962.68497	1016.4372	1016.522962	1118.279350
Smurf	12590	197.478455	147.136010	0.977958	23.161680	328.063862	328.264040	658.090443

Class	Count	Mean	std	min	25%	50%	75%	max
SIDDOS	6665	104.976245	48.120505	0.992637	94.7212	94.7212	94.7212	658.090443

Table 2 Statistical description of the dataset for 'BYTE_RATE'

Class	Count	Mean	std	min	25%	50%	75%	max
Normal	1935959	232456.9	224611.7	53.78760	18056.40	124942	505437	1438057
HTTP-Flood	4110	725682.3	459193.2	5209.670	325885.9	726208.8	1130920	1519216
UDP-Flood	201344	1271845	452197.7	53.78770	1.147520	1524630	1524780	1677420
Smurf	12590	630728.1	630263.3	53.78770	18067	505434	1509180	1521430
SIDDOS	6665	18595.943463	76993.119007	1528.66	5209.67	5209.67	5209.67	505985

To reduce the number of features, the attributes are analyzed using the Chi-Square test, which is shown in the equation:

$$x_c^2 = \frac{\sum(0_i - E_i)^2}{E_i} \quad (1)$$

where χ_c^2 is the Chi-square value, O_i is the Number of observations in class i, and E_i is the number of expected observations in class i if there is no relationship between the feature and the target. Thus, the Chi-square test works by finding the relationship between the independent attribute (predictor values) and the dependant attribute (target value). A higher Chi-square score indicates a higher relationship between the dependent and independent attribute, meaning that a change in the predictor value will influence a change in the target value. The results of the chi-square analysis are shown in *Figure 1*, through which it is evident that the attributes "Packet Rate" and "Byte Rate" are the most important features for determining whether a packet is normal traffic or attacking traffic. The 3D visualization of the modified dataset can be seen in *Figure 2*, where there is a clear distinction between normal packets and attacking packets. The two parameters chosen from the dataset are PACKET_RATE and BYTE_RATE. It is observed that as the value of these two parameters increases, so does the probability that the packet is of attacking traffic. The Chi-square values of the top five parameters are given below in *Table 3*.

Table 3 Top 5 Chi-square values of parameters

Parameter	Chi-Square value
PACKET_RATE	0.20814693
BYTE_RATE	0.15693363
FLAGS	0.08422273
PKT_SEND_TIME	0.05499739

4.3Training model

A comparison of various Machine learning models is performed to determine a model that would give the most accurate results while taking the least amount of time to train. The different models which have been experimented with are CatBoost Classifier, XGBoost classifier, logistic regression, Naive Bayes classifier, Decision Tree classifier, and KNN classifier.

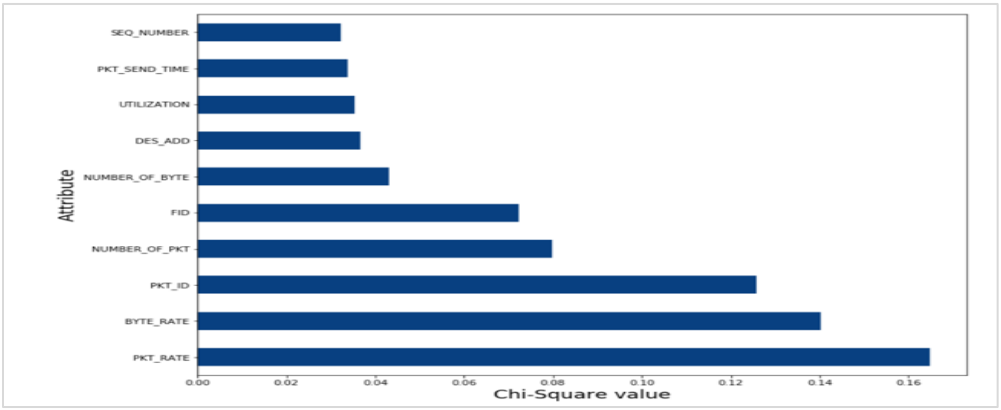

Figure 1 Chi-Square test results

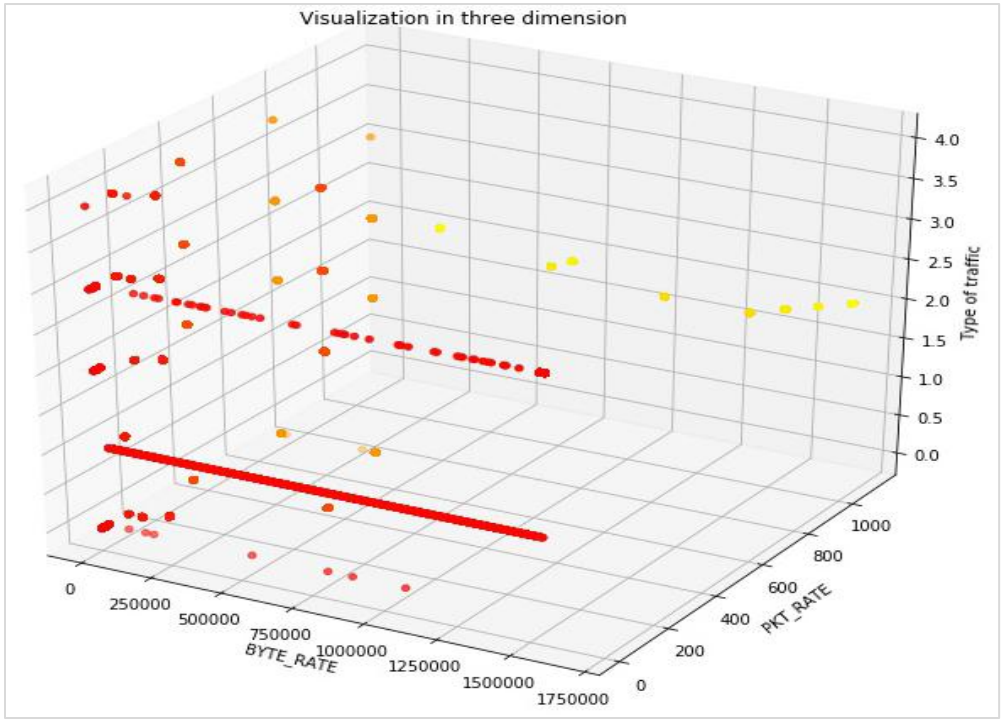

Figure 2 3D visualization of dataset

Both the Boosting algorithms have been trained on 100 epochs. The performance of each of the models is measured using different metrics such as accuracy, precision, recall. Training time has also been used as a metric for analysis as DDoS attacks need to be identified as fast as possible to prevent loss to the organization. The Matplotlib library of python is used to visualize the results for better understanding. The Loss vs Epoch during the training of the CatBoost classifier is shown in *Figure 3* and Loss vs Epoch during cross-validation is shown in *Figure 4*. It can be seen that the model reaches the local minima 10 epochs into training whereas it takes 90 epochs during cross-validation.

Figure 3 Training loss vs epoch graph

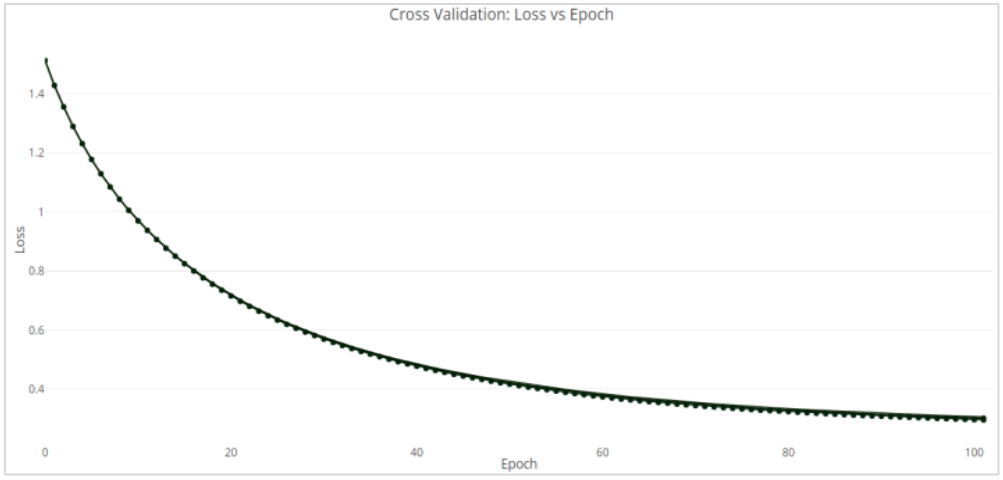

Figure 4 Cross-Validation loss vs epoch graph

4.4Setting up SDN environment

Mininet is used to simulate the SDN environment where DDoS attack, its detection, and mitigation are tested. A topology is designed as shown in *Figure 5*. The SDN network designed for this experiment is a standard TreeNet topology available in Mininet. This topology has two parameters depth and fanout which have been set to 3 and 2 respectively. The depth indicates the number of levels of switches that will be present in the topology and the fanout is the number of child nodes each parent node would have. The first three layers consist of a total of 7 switches and the last layer consists of 8 hosts, each switch in the penultimate layer having 2 hosts each. A single Ryu controller is connected to the root switch. In the experiment, the host h1 (IP 10.0.0.1) will act as the botmaster, host h3(IP 10.0.0.3) will act as the victim host, and hosts h7 (10.0.0.7) and h8 (10.0.0.8) act as the botnet. For a controller in the SDN network, RYU is used which is a python-based controller. RYU's northbound APIs which are in the form of rest APIs provide flow information between DataPath and enable deployment or modification of flow rules in the SDN network. The packet rate and byte rate after being retrieved from the APIs is given as input to the machine learning model which classifies the flow as normal traffic or DDoS attack. Figure 6 shows the block diagram of the proposed solution.

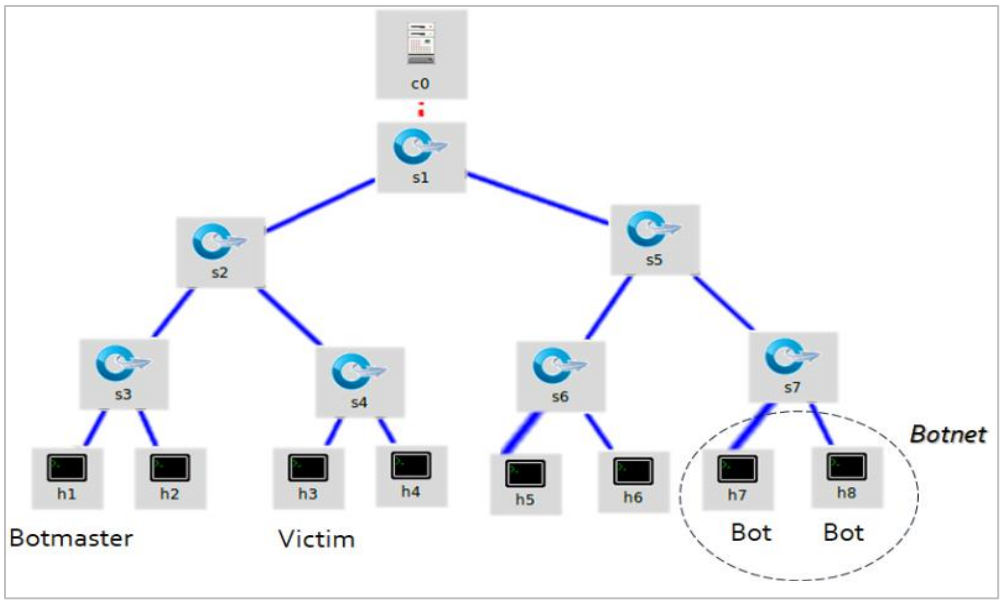

Figure 5 SDN topology

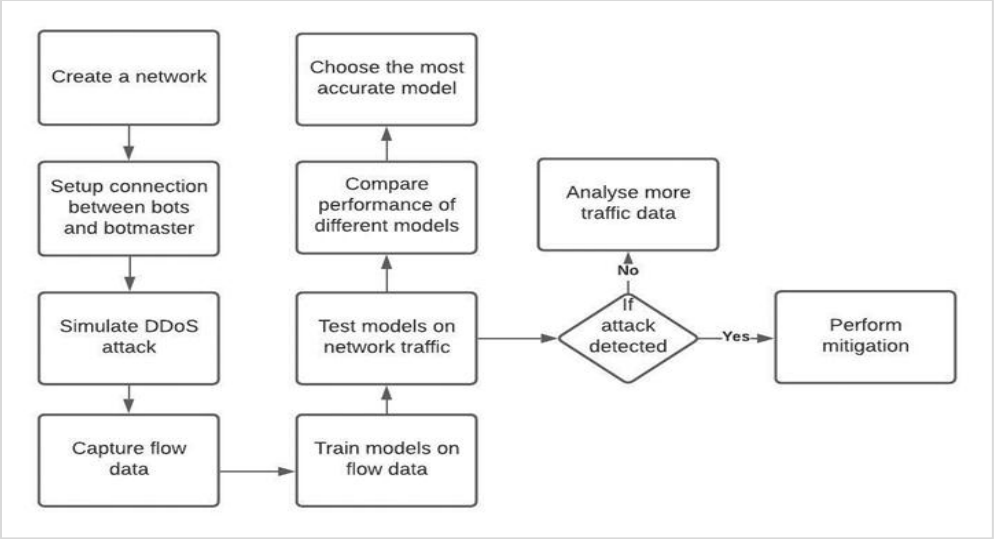

Figure 6 Block diagram of the proposed Solution

4.5Connection setup and traffic generation

The botmaster executed a script that took control over the botnets using an SSH connection. The command-line tool hping3 is used to generate both normal and DDoS UDP traffic. Normal traffic in this experiment is the traffic generated to represent the traffic generated by a non-malicious user on the network. First, normal traffic is generated and then the victim is bombarded with UDP flood traffic by the two SSH botnets. A DDoS attack is simulated and the change in flow rate is observed and analyzed through Wireshark.

4.6Detection and mitigation

For detection and mitigation of attack, an SDN application running over the controller is written in python which will communicate with the controller and retrieve required flow stats i.e., packet rate and byte rate along with the source and destination mac addresses of the flow at the desired interface. The retrieved data is fed to the model as input in a continuous manner to detect the attack. After an attack is detected by the ML model, the already fetched source and destination mac addresses between which the DDoS traffic was observed, are used as the match parameters to deploy the flow rules onto the switch using the RYU Northbound APIs, thus preventing further communication between the hosts.

5.Results

The DDoS attack in Mininet is simulated. Normal traffic is simulated until the 75th second after which the DDoS attack starts. The sudden change in the packet rate signifies the start of the DDoS attack by the botnet. The packets/sec vs time is plotted for the whole duration of the experiment and is shown in *Figure 7*.

The detector-mitigator module is now added and the same experiment is simulated in the same conditions. This time too, normal traffic is generated until the 75th second and then the DDoS attack starts. The detector-mitigator module can detect the DDoS attack and can deploy the necessary flow rules to mitigate the attack. The sudden decrease in packet rate signifies the mitigation of the DDoS attack due to the deployment of the flow rules. The packets/sec vs time is plotted for the whole duration of the experiment and is shown in *Figure 8*.

When the DDoS attack is initiated, the detector-mitigator SDN application running on top of the controller retrieves the packet rate and byte rate from the Ryu Northbound APIs and passes them as input to the machine learning model in the SDN application for detection of DDoS attack. When the attack is detected, the mitigator part deploys flow rules to disable the connection between the botnet and the victim through the Ryu Northbound APIs. The sudden decrease in packet rate observed in the graph is due to the blocking of connection between victim and botnet after detection. Two peaks are observed as there are 2 hosts in the botnet and there is a minute time gap during the detection and mitigation of the attack between the 2 hosts in the botnet

The detector-mitigator SDN application running on the controller takes in packet rate and byte rate as input and passes them as input to the machine learning model. If a DDoS attack is detected then the following message is printed as shown in *Figure 9.*

Once a DDoS attack is detected, flow rules are deployed by the SDN application to the required switches using the RYU Northbound APIs. The last two rules in *Figure 10* are the rules deployed to mitigate the attacks.

Once the flow rules are deployed the communication between the bots in the botnet and the victim host is completely blocked. It is evident when the 'pinball command is used. All the connections exist except the connection between the botnet and the victim host. This is shown in *Figure 11*.

Six Machine Learning models were tested based on the literature survey done namely the Catboost model, XGBoost model [2,19], K-Nearest Neighbour Model [9], Logistic Regression Model [10], Decision Tree model [12,14], and Gaussian Naive Bayes Model [14].

It is seen in *Figure 12* that Catboost, XGBoost, and K-Nearest Neighbours all gave the highest accuracy of 98% followed by Gaussian Naive Bayes with an accuracy of 97%, followed by Decision Trees and Logistic Regression which gave the lowest accuracy of 89%.

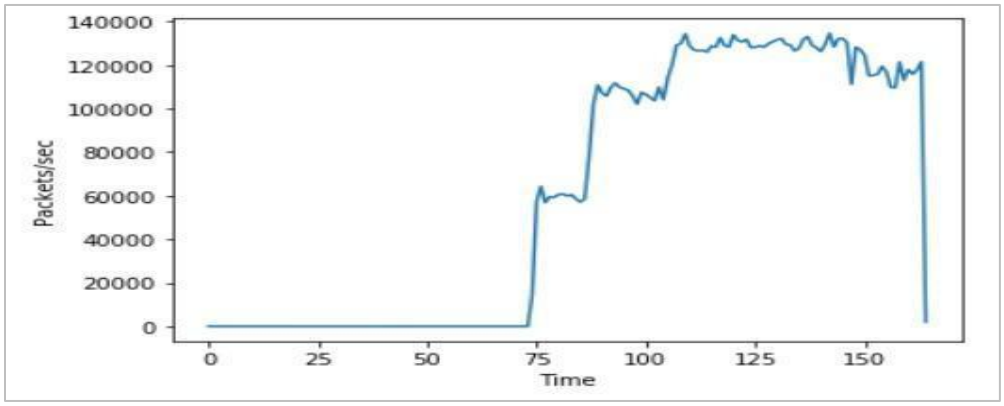

Figure 7 Packets/sec vs time graph at the victim host interface

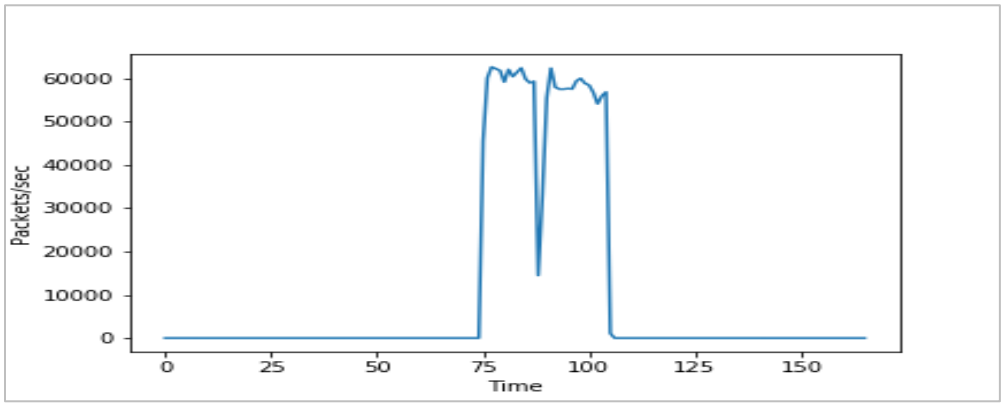

Figure 8 Packets/sec vs time graph at victim host interface after deploying the SDN application

```
Src:9e:0d:f4:30:43:17 and dest: 0e:0b:d6:cd:38:9e DDoS
flow rules added

Src:4e:83:8f:78:3f:b4 and dest: 0e:0b:d6:cd:38:9e DDoS
flow rules added
```

Figure 9 Output of the detector-mitigator SDN application written in python

```
mininet> sh ovs-ofctl dump-flows s4
 cookie=0x0, duration=226.765s, table=0, n_packets=4, n_bytes=286, in_port="s4-eth1",dl_src=0e:0b:d6:cd:38:9e,dl_dst=1a:60:7f:05:34:a9 actions=output:"s4-eth3"
 cookie=0x0, duration=226.757s, table=0, n_packets=2, n_bytes=96, in_port="s4-eth3",dl_src=1a:60:7f:05:34:a9,dl_dst=0e:0b:d6:cd:38:9e actions=output:"s4-eth1"
 cookie=0x0, duration=226.746s, table=0, n_packets=4, n_bytes=286, in_port="s4-eth2",dl_src=3e:82:2a:8b:18:1b,dl_dst=1a:60:7f:05:34:a9 actions=output:"s4-eth3"
 cookie=0x0, duration=226.738s, table=0, n_packets=4, n_bytes=228, in_port="s4-eth3",dl_src=1a:60:7f:05:34:a9,dl_dst=3e:82:2a:8b:18:1b actions=output:"s4-eth2"
 cookie=0x0, duration=188.503s, table=0, n_packets=180, n_bytes=12320, in_port="s4-eth1",dl_src=0e:0b:d6:cd:38:9e,dl_dst=c6:5d:8b:e8:8f:89 actions=output:"s4-eth3"
 cookie=0x0, duration=188.489s, table=0, n_packets=386, n_bytes=16212, in_port="s4-eth3",dl_src=c6:5d:8b:e8:8f:89,dl_dst=0e:0b:d6:cd:38:9e actions=output:"s4-eth1"
 cookie=0x0, duration=113.857s, table=0, n_packets=22, n_bytes=1456, in_port="s4-eth1",dl_src=0e:0b:d6:cd:38:9e,dl_dst=4e:83:8f:78:3f:b4 actions=output:"s4-eth3"
 cookie=0x0, duration=99.834s, table=0, n_packets=24, n_bytes=1596, in_port="s4-eth1",dl_src=0e:0b:d6:cd:38:9e,dl_dst=9e:0d:f4:30:43:17 actions=output:"s4-eth3"
 cookie=0x1, duration=101.711s, table=0, n_packets=6907292, n_bytes=290106264, in_port="s4-eth3",dl_src=4e:83:8f:78:3f:b4,dl_dst=0e:0b:d6:cd:38:9e actions=drop
 cookie=0x1, duration=83.992s, table=0, n_packets=5696857, n_bytes=239267994, in_port="s4-eth3",dl_src=9e:0d:f4:30:43:17,dl_dst=0e:0b:d6:cd:38:9e actions=drop
```

Figure 10 Flow rules added at the necessary switch

```
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 X X
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 X h4 h5 h6 h8
h8 -> h1 h2 X h4 h5 h6 h7
*** Results: 7% dropped (52/56 received)
```

Figure 11 Result of pingall command

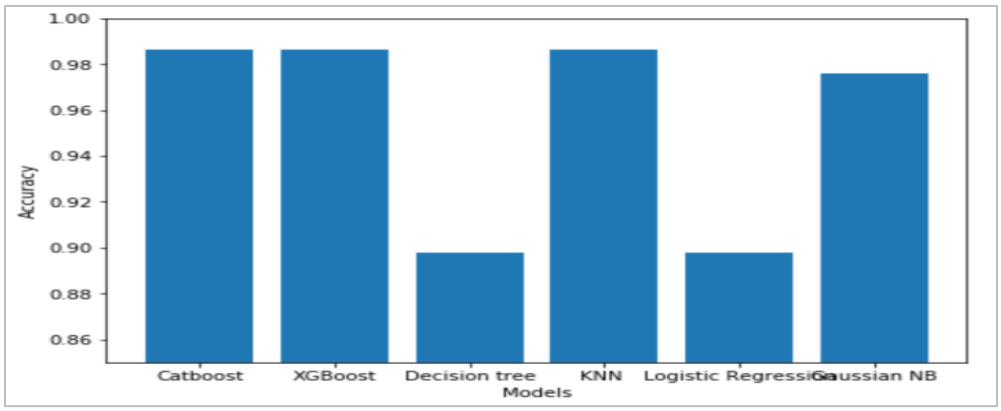

Figure 12 Accuracy of the models tested

The training time for all the six models is also observed and it is seen in *Figure 13* that XGBoost had the highest training time which is undesirable, followed by K-Nearest Neighbour, followed by Catboost and Logistic Regression which roughly had the same time, followed by Decision Tree and finally Gaussian Naive Bayes which had the least training time. On account of both accuracy and training time, it is evident that Catboost is the best model to choose from the current set due to its very high accuracy and low training time when compared to other models with similar high accuracies. The values of Accuracy and Training time for all of the models is given in *Table 4.* It is seen in *Table 4* that the highest accuracy of 98.62% is obtained by using the CatBoost classifier, as well as other classifiers such as XGBoost and KNN. However, among the models with the highest accuracy, it is observed that the CatBoost classifier has the least training time of 119 seconds, followed by 149 seconds for the KNN classifier. XGBoost has the highest training time of 658.5 seconds, which is almost 5.5 times the training time of CatBoost. The table also provides other measures of performance such as precision, recall, and F1-score. It is observed that CatBoost, XGBoost, and KNN have the best values of precision and F1-score. XGBoost has a better Recall score than the other two models by 0.01.

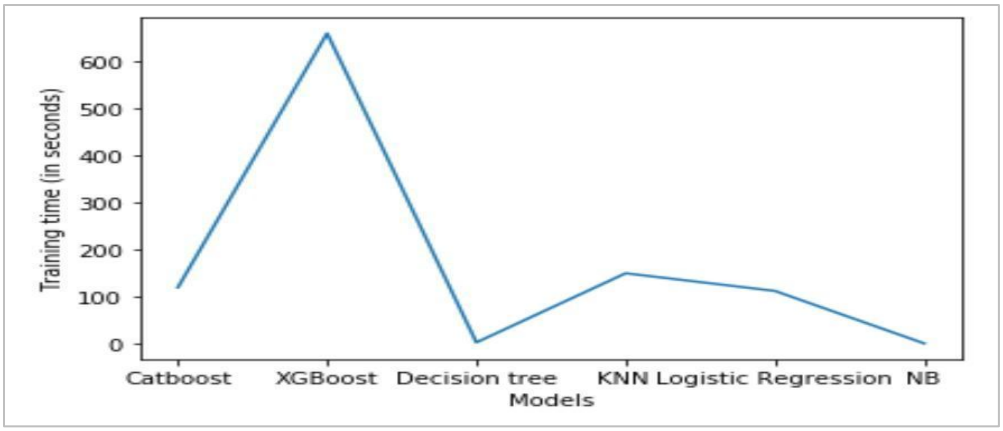

Figure 13 Training time of the model tested

Table 4 Quantitative comparison of Accuracy and Training time of all the models experimented

ML Model	Accuracy	Precision	Recall	F1 - Score	Training Time (seconds)
CatBoost	0.9862	0.97	0.82	0.86	119
XGBoost [2,19]	0.9862	0.97	0.83	0.86	658.571
Decision Tree [12,14]	0.8980	0.29	0.26	0.27	
	2.293				
K Nearest Neighbours (KNN) [9]	0.9862	0.97	0.82	0.86	
	149.159				
Logistic regression [10]	0.8980	0.29	0.26	0.27	
	111.693				
Gaussian Naive Bayes [14]	0.9757	0.46	0.50		
	0.47	0.273			

6.Discussion

Catboost algorithm, launched in 2017 is a relatively new algorithm that provides various features like fast prediction, improved accuracy, a requirement of minimal hyperparameter tuning, etc. when compared to its counterpart. The lack of its usage in the networking domain prompted the team to test this algorithm out in detecting DDoS attacks in an SDN environment. From the results, it is evident that while the accuracy of the Catboost algorithm is similar to KNN and XGBoost, the training time is less when compared to KNN and XGBoost. There are certain limitations to the approach and the experimentation performed. The first being that two parameters packet rate and byte rate are considered for analysis. There could be more parameters that might affect the accuracy but due to limitations of the API used, parameters that can provide optimal results among all available parameters are considered. Another limitation can be the cases of false positives where some genuine host with a large number of requests might be considered to be an attacker and blocked from further communication. The experiment was carried out on RYU controller as it provides northbound APIs which delivered the parameters required for the analysis. Hence, some other controller might not be able to fulfill the objective of the experiment. Also, the analysis was only performed on UDP packets hence it can be further improved by testing more types of network packets.

7.Conclusion and future scope

In this experiment, a DDoS attack is simulated in an SDN environment using an SSH botnet. To detect and mitigate such an attack, an SDN application that runs on the controller is written in python to detect the attack using a machine learning model and mitigate the attack using flow rules. For the detection part, six models are used namely Catboost, XGBoost, Logistic Regression, Gaussian Naive Bayes, and

Decision Tree. it is observed that the Catboost model, despite having the same accuracy as the XGBoost model and K-Nearest Neighbour model, is chosen as the better model when compared to the rest on account of its training time. The future scope of this research includes,

1. Testing the detector-mitigator module for other types of attacks such as HyperText Transfer Protocol flood attack, Ping flood attacks, etc.
2. Increasing the number of attributes used for prediction by the detector module to increase the accuracy.
3. Detection of the botmaster and limiting its activities in the network

Acknowledgment
None.

Conflicts of interest
The authors have no conflicts of interest to declare.

References
[1] Chen Z, Jiang F, Cheng Y, Gu X, Liu W, Peng J. XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud. In international conference on big data and smart computing (bigcomp) 2018 (pp. 251-6). IEEE.
[2] Thomas RM, James D. DDOS detection and denial using third party application in SDN. In international conference on energy, communication, data analytics and soft computing 2017 (pp. 3892-7). IEEE.
[3] Lukaseder T, Stölzle K, Kleber S, Erb B, Kargl F. An SDN-based approach for defending against reflective ddos attacks. In conference on local computer networks 2018 (pp. 299-302). IEEE.
[4] Hong K, Kim Y, Choi H, Park J. SDN-assisted slow HTTP DDoS attack defense method. IEEE Communications Letters. 2017; 22(4):688-91.
[5] Deepa V, Sudar KM, Deepalakshmi P. Detection of DDoS attack on SDN control plane using hybrid machine learning techniques. In international conference on smart systems and inventive technology 2018 (pp. 299-303). IEEE.
[6] Lawal BH, Nuray AT. Real-time detection and mitigation of distributed denial of service (DDoS) attacks in software defined networking (SDN). In signal processing and communications applications conference 2018 (pp. 1-4). IEEE.
[7] Wijesinghe U, Tupakula U, Varadharajan V. Botnet detection using software defined networking. In international conference on telecommunications 2015 (pp. 219-24). IEEE.
[8] Dao NN, Park J, Park M, Cho S. A feasible method to combat against DDoS attack in SDN network. In international conference on information networking 2015 (pp. 309-11). IEEE.
[9] Dong S, Sarem M. DDoS attack detection method based on improved KNN with the degree of DDoS attack in software-defined networks. IEEE Access. 2019; 8:5039-48.
[10] Yadav S, Selvakumar S. Detection of application layer DDoS attack by modeling user behavior using logistic regression. In international conference on reliability, infocom technologies and optimization (Trends and Future Directions) 2015 (pp. 1-6). IEEE.
[11] Fouladi RF, Kayatas CE, Anarim E. Frequency based DDoS attack detection approach using naive Bayes classification. In international conference on telecommunications and signal processing 2016 (pp. 104-7). IEEE.
[12] Lakshminarasimman S, Ruswin S, Sundarakantham K. Detecting DDoS attacks using decision tree algorithm. In fourth international conference on signal processing, communication and networking 2017 (pp. 1-6). IEEE.
[13] Sahoo KS, Tripathy BK, Naik K, Ramasubbareddy S, Balusamy B, Khari M, et al. An evolutionary SVM model for DDOS attack detection in software defined networks. IEEE Access. 2020;8: 132502-13.
[14] Pérez-Díaz JA, Valdovinos IA, Choo KK, Zhu D. A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. IEEE Access. 2020; 8:155859-72.
[15] Abou El Houda Z, Khoukhi L, Hafid AS. Bringing intelligence to software defined networks: mitigating DDoS Attacks. IEEE Transactions on Network and Service Management. 2020; 17(4):2523-35.
[16] Gong C, Yu D, Zhao L, Li X, Li X. An intelligent trust model for hybrid DDoS detection in software defined networks. Concurrency and Computation: Practice and Experience. 2020; 32(16):e5264.
[17] Phan TV, Park M. Efficient distributed denial-of-service attack defense in SDN-based cloud. IEEE Access. 2019; 7:18701-14.
[18] Tan L, Pan Y, Wu J, Zhou J, Jiang H, Deng Y. A new framework for DDoS attack detection and defense in SDN environment. IEEE Access. 2020; 8:161908-19.
[19] Alamri HA, Thayananthan V. Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against DDoS attacks. IEEE Access. 2020; 8:194269-88.
[20] Wang J, Wen R, Li J, Yan F, Zhao B, Yu F. Detecting and mitigating target link-flooding attacks using SDN. IEEE Transactions on Dependable and Secure Computing. 2018; 16(6):944-56.
[21] Jia Y, Zhong F, Alrawais A, Gong B, Cheng X. Flowguard: an intelligent edge defense mechanism against IoT DDoS attacks. IEEE Internet of Things Journal. 2020; 7(10):9552-62.

Sanjeetha R, is a Research Scholar and Assistant Professor is working in the Department of Computer Science and Engineering at M S Ramaiah Institute of Technology. She is a senior IEEE member and life member of ISTE. Her area of interests includes Software Defined Networks, Computer Networks and Data Communications
Email: sanjeetha.r@msrit.edu

Anant Raj, is currently pursuing his Bachelor's degree in Computer Science and Engineering in MS Ramaiah Institute of Technology, Bangalore and will graduate in 2021. He is also working as an intern in JP Morgan Chase & Co. His fields of interest are Machine Learning, Software Defined Networks and Web & App Development.
Email: anant.rj.421@gmail.com

Kolli Saivenu, is pursuing his Bachelors of Engineering in Computer Science and Engineering from MS Ramaiah Institute of Technology and will graduate in the year 2021. His fields of interest are Machine Learning, Deep Learning and Software Defined Networks.
Email: ksaivenu2010@gmail.com

Mumtaz Irteqa Ahmed, is pursuing his Bachelors of Engineering in Computer Science and Engineering from MS Ramaiah Institute of Technology and will graduate in the year 2021. His areas of interests include Software Development, Machine Learning and Network Security.
Email: fareed2000ahmed@gmail.com

Sathvik B, is pursuing his Bachelor's degree in Computer Science and Engineering in MS Ramaiah Institute of Technology, Bangalore, graduating in 2021. He is a member of IEEE and is working as an Intern in JP Morgan Chase & Co. His are of interest include Machine Learning and Software Defined Networks.
Email: sathvikbk123@gmail.com

Dr. Anita Kanavalli received her Ph.D. in Computer Science and Engineering from Bangalore University in 2013. She is a member of Indian Society of Technical Education (ISTE) and IEEE. She is also a certified instructor for CISCO Networking Academy for CCNA.
Email: anithak@msrit.edu