ISSN 2320-9984
Volume 10 Issue 3 | September 2022 ISSN: 2320-9984 (Online)
International Journal of Modern Engineering & Management Research
Website: www.ijmemr.org

A Machine Learning Framework for Detection of Distributed Denial of Service Attack Using Random Forest Algorithm

Prathibha Kumari
M.Tech. Research Scholar
Computer Science and Engineering
Shriram Group of Institutions
Jabalpur, (M.P.) India
Email: kumaripratibha734@gmail.com

Sapna Jain Choudhary
Assistant Professor
Department of Computer Science and Engineering
Shriram Group of Institutions
Jabalpur, (M.P.) India
Email: choudharysapnajain@gmail.com

Abstract*—Denial of Distribution of Service (DDoS) attack make the challenges to provide the large data centers are storing and sending more and more data. In order to check whether the network traffic does not contain intrusions, an intrusion detection system is used. The objectives of this study are to introduce machine learning concepts and Random Forest algorithm for Denial of Distribution of Service. The algorithms are evaluated on different data sets, which consist of both labeled training data and unlabeled real world data. In the evaluation, it is found that supervised learning gives better and more detailed predictions as compared to unsupervised learning. The results show that machine learning is a viable option to detect intrusions using IP flows.*

Keywords:*—Cloud Security, DDoS Attack, Cloud Virtualization, Virtual Machine & Random Forest algorithm.*

1. INTRODUCTION

A denial-of-service (DoS) attack is a malicious attempt to overwhelm an online service and render it unusable. One of the major network attacks is a Denial of Distribution of Service (DDoS) attack. With the immense internet growth, a large number of hosts are vulnerable to the attacks. Ensuring security in SDN is very important to provide secure communication. This research concentrates on Distributed Denial of Service Attack (DDoS) on the data plane. The DDoS attack makes a machine or network resources unavailable to its users. This is achieved by consuming the entire network bandwidth or the resources of the network nodes (such as memory and CPU). Figure 1 shows the DDoS attack on an SDN controller.

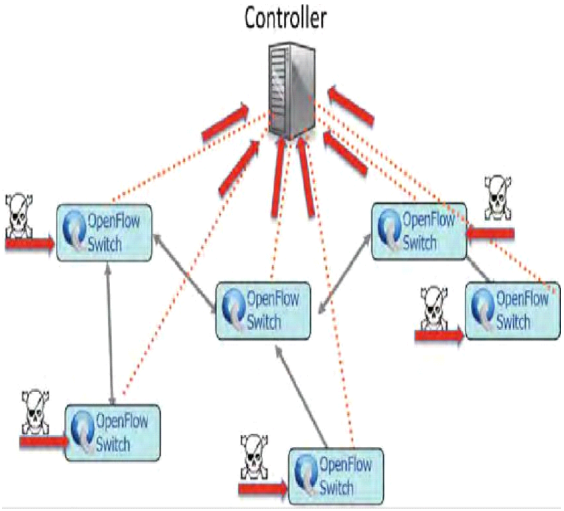

Figure 1: DDoS Attack on SDN controller

1.1 SDN controller

The controller is considered as the core of an SDN network. The controller uses protocols like OpenFlow to communicate with networking devices. There are different types of controllers like POX, Ryu, Open Day Light, Beacon, etc. The Figure 1 shows the different SDN controllers. In this research, POX controller is used.

1.2 POX

POX is inherited from NOX controller [5]. It is an open source development platform, used to create an SDN controller using python programming language. POX controller provides an efficient way to handle the OpenFlow devices. Using the POX controller, you can run different applications like a hub, switch, load balancer, and firewall. It is a great tool for SDN research works. The proposed algorithm in this research is implemented in the pox controller.

	POX	Ryu	Trema	Floodlight	Open Day Light
Language Support	Python	Python	C Ruby	Java	Java
OpenFlow Support	v1.0	v1.0 v1.2 v1.3	v1.0	v1.0	v1.0
OpenSource	Yes	Yes	Yes	Yes	Yes
GUI	Yes	Yes	No	Web GUI	Yes
REST API	No	Yes	No	Yes	Yes
Platform Support	Linux Mac Windows	Linux	Linux	Linux	Linux Mac Windows

Figure 2: Types of SDN controller

1.3 Type of DDoS Attack

UDP Flood [6] is a type of attacks which aims at bringing down the server by sending a large number of UDP packets to random ports on the targeted host. The attackers usually utilize the UDP's connectionless feature to submit a stream of UDP data packet to the victim machine. The victim machine's queue becomes filled and it will not be able to respond legitimate user's request. Usually, in these types of attacks, the attacker spoofs the source IP address of the UDP packets to hide the locations of the attack machines.

SYN Flood is an attack using TCP connection initiation to target the victim's machine. A large number of SYN packets are sent to the victim but no ACK is returned to the victim, causing a large number of resources at the victim's machine and making the machine unavailable to legitimate users.

2. MACHINE LEARNING [1]

Machine learning is an Artificial Intelligence application that provides the computer program the ability to learn from input data [8]. It allows us to use historical data as the input for predicting future data. Thus, the accuracy of the output is solely based on the quality of the historical data. Nowadays, machine learning techniques are used in various fields to solve different problems. For example, they are used in Email spam filtering, pattern and image recognition, search engines filtering, healthcare applications, etc.

2.1 Types of Machine learning algorithms

Machine learning algorithms can be broadly classified as Supervised learning algorithms and Unsupervised learning algorithms.

Machine learning algorithms can be broadly classified as **Supervised learning algorithms** and **Unsupervised learning algorithms.**

2.2 Supervised Learning algorithms

A Supervised learning algorithm is mainly used to solve classification and regression problems as it makes the detection or decision-making process easier. It uses the past learned data to predict the future events. The input data used to train the learning algorithm is a labeled one. That is, the input data have one or more labels, e.g. in our thesis attack and no attack are the labels used to classify the traffic data. After appropriate training, the system can classify unknown data. This research uses supervised learning algorithms.

2.3 Unsupervised learning Algorithms

Unsupervised learning algorithm uses unlabeled input data to train the system. That is, the input data are not tagged with labels. It finds the hidden structure from the unlabeled input and groups them as clusters showing the similarities. The initial performance of this type of learning algorithm is poor, but the system can tune itself to improve the performance.

2.4 Supervised Machine learning

This is the most commonly used technique in machine learning. Our research problem implements a Supervised machine learning algorithm to classify the network traffic as legitimate traffic and malicious traffic. Here the classifier gets the input which is a set of feature values also called input vector and outputs the predicted value called class. Figure 3 shows the supervised learning classifier.

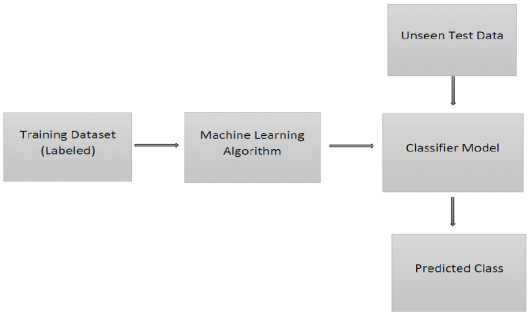

Figure 3: *Supervised Learning classifier*

Here the training data are given as an input to the learning algorithm which results in a classifier model. The performance of the classifier can be evaluated using unseen data.

3. SOFTWARE DEFINED NETWORKING

SDN is an emerging technology, which enables the network to be programmable, centralized and flexible. The SDN architecture has a separate control plane and data plane. System administrators can control the entire network through the centralized control plane (controller). SDN is an emerging network architecture which is dynamic, manageable and cost-effective.

It is based on the abstraction of forwarding plane from the control plane. This abstraction makes the network directly programmable and flexible, which is ideal for configuring, managing, securing and optimizing the network resources dynamically and automatically. These features of SDN can be used in the construction of intelligent and automated networks. Also, the operational costs in large data centers**.**

SDN can make decisions about how packets should flow through the network in the forwarding plane. Packet handling rules are sent to the switches from a controller. The controller is a software application running on a server located remotely. The switches seek guidance from the controller for packet handling.

Switches and controller communicate via the controller's south-bound interface. This communication is achieved by the OpenFlow protocol. Similarly, applications can talk to the controller via the controller's north-bound interface. The SDN architecture is shown in Figure 4

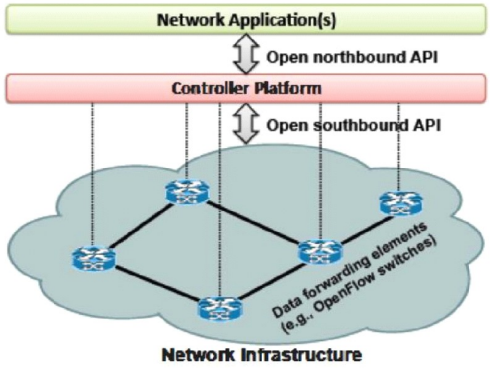

Figure 4: Simplified view of SDN Architecture

3.1. Benefits of SDN

SDN facilitates automated load balancing and it has the ability to scale network resources dynamically. The open standard implementation simplifies the network design and operations. It is ideal for today's high-bandwidth applications.

Today it's easier to unify cloud resources with SDN. Large data center platforms can be easily managed from SDN controller. It's also used to implement centralized security.

3.2 OpenFlow Protocol

The communication between the controller and the data plane devices needs a suitable standard. OpenFlow is the communication interface defined between the control and forwarding layers of an SDN architecture [3]. OpenFlow manages the switches in the network and allows the controller to manipulate the flow of packets through the network.

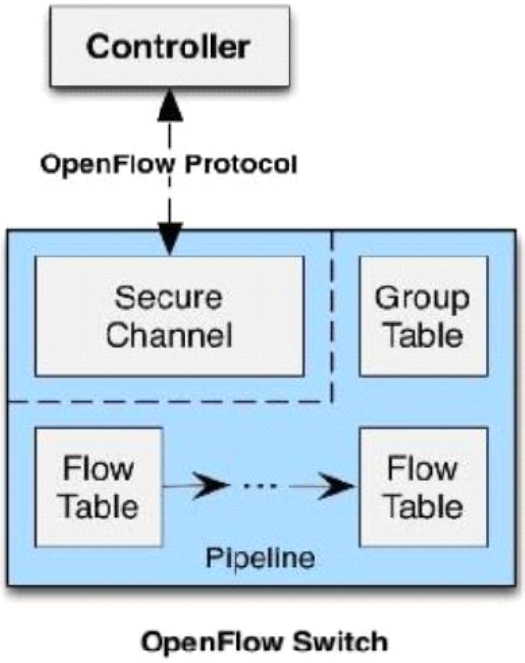

Figure 5: OpenFlow switch

Figure 5 shows the structure of an OpenFlow switch. An OpenFlow switch consists of one or more flow tables, a group table and a secure channel to an external controller. The switch communicates with the controller, and the controller manages the switch via OpenFlow protocol. Each flow table in the switch contains a set of flow entries. Using the OpenFlow protocol, the controller can add, delete and update flow entries in the flow table.

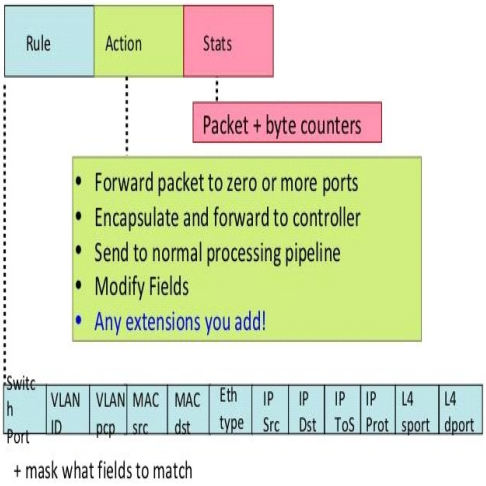

Figure 6: *Flow Table Entries*

Random forest is one of the most powerful algorithms used for predictive modeling [8]. The underlying principle is the construction of multiple decision trees by randomizing the combination of variables. That is, multiple decision trees are constructed from the given data set and the results are combined to make predictions. To construct multiple decision trees, the data set is divided repeatedly into subtrees by changing the combination of variables. The challenge here is to find the best combination of variables which gives the highest accuracy in prediction.

The accuracy of the Random Forest algorithm can be tuned by increasing the number of trees generated. Each individual decision tree generated makes its own prediction. Some may be right and some wrong. The individual trees that produced correct predictions reinforce each other, while wrong predictions get canceled. For this to happen, the individual trees generated must be uncorrelated. Here comes the Bagging technique which helps in generating the decision trees with minimal correlation.

Random Forest is an ensemble classifier which can implement Bootstrap aggregation (Bagging) to improve the accuracy. That is, normally the learning algorithm can choose the split point from all the available features. But the Random Forest algorithm which

implements bagging technique, while constructing the individual trees, randomly chooses a node to split on.

For example, if a dataset had 16 input variables for a classification problem, then the randomly selected features x is given by,

$$x = \sqrt{16}$$

$$x = 4$$

Thus, the individual decision trees are constructed based on the randomly selected 4 features.

Figure 7 shows the process involved in RF algorithm to create decision trees and, to derive the predictions out of them. Here the training dataset D has d1, d2, ...dN variables. Using Bootstrap process, it creates m decision trees. The average value of the results obtained in each decision tree is calculated and the result is predicted.

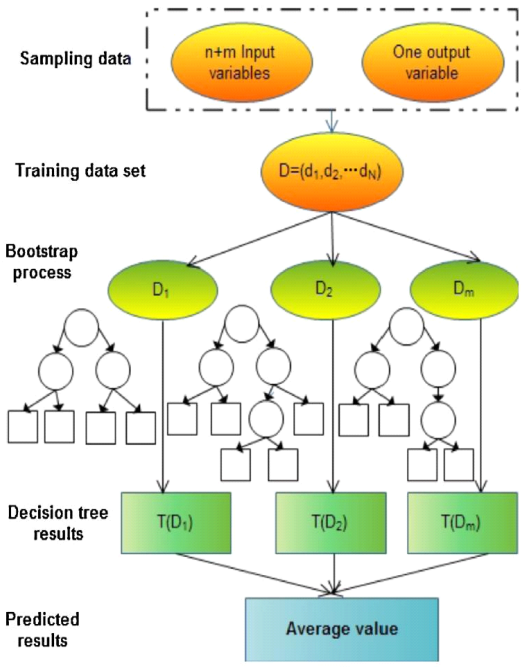

Figure 7: *Random Forest Algorithm*

The process of creating a single decision tree can be explained with an example [39]. Figure 8 shows a dataset of people who buy a computer.

3.3. How Random Forest Works:

Random Forest Creation Algorithm

1. Let the number of training cases in the dataset be N.
2. Select randomly *m* features from total *M* features, such that m is much less than *M*.
3. Among m features calculate the node d, using the best split point.
4. Split the d node into daughter nodes.
5. Repeat 2 to 4 until a leaf node has been reached.
6. Build the forest by repeating steps 2 to 5 for i number of times. The value of i is equal to the number of trees to be created.

3.4 Random Forest Prediction Algorithm

After the creation of forest using the training dataset, we can perform prediction for the unknown test data using Majority voting.

1. Select the test features and use the rules of each randomly created decision tree to predict the result. Save the result as the target.
2. Calculate the votes for each predicted target.
3. High voted target is declared as final prediction

Internally random forest creates many independent decision trees and sets the rules for each decision tree based on the values of the input variables. There is no need to set the classification rules manually. So the dataset plays an important role here. To get accurate results, our datasets should be error free.

The process of creating a single decision tree can be explained with an example [39]. Figure 8 shows a dataset of people who buy a computer.

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

***Figure 8**: Example Dataset of people who buys computer*

In Figure 9 the dataset is divided into 3 subsets based on the attribute "age".

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

***Figure 9:** Dataset divided into 3 Subsets*

The attribute age has three different values: <=30, 31...40, >40. From the table, we can learn that all students with age <=30 buys a computer. The people with age = 31...40 buys a computer. Also, people with age >40 and fair credit rating buy a computer. Based on these analyses the decision tree is generated.

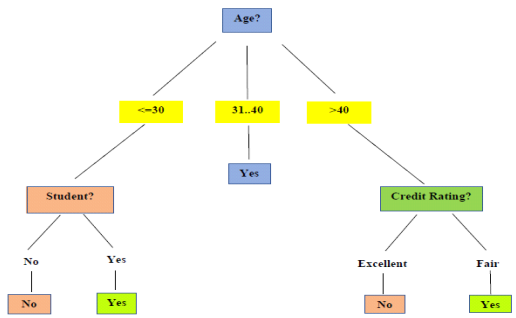

***Figure 10:** Single Decision Tree*

3.5 Extracting Classification Rules

Each attribute-value pair along a path from the root to leaf forms a rule. The leaf node holds the class prediction. For e.g.

If age = "<=30" and student = "no" then buys_computer = "no"

If age = "<=30" and student = "yes" then buys_computer = "yes"

Consider a new data: (<=30, yes, excellent, ?). To find the class value of new data, the tree is analyzed, and the class value is computed as a yes.

3.6 Feature Selection

Though there may be numerous features available in the given dataset, only features relevant to the problem to be solved are selected. This is called feature selection. To select the relevant features, we need to find the importance of each feature in predicting the results. This is done by calculating how much the error rate drops for a feature at each split point. Those features with small error rate are considered as more important for the classification problem. This is called a Gini score. The Gini score of a feature can be calculated by the following formula,

$$G\,i\,n\,i = 1 - n \sum i = 1\,(\,p\,i\,)\,2$$

3.7 Classifier Accuracy Estimation

Estimation of the predictive accuracy of a classifier is done to know how good the prediction will be. Common methods used for accuracy estimation are [40]: Validation set approach and k-fold cross-validation.

In validation set approach, to measure the classifier accuracy, the dataset is divided into a training dataset (50%) and testing dataset (50%). Once the classifier model is built using the training dataset, the accuracy is estimated using the unused testing dataset.

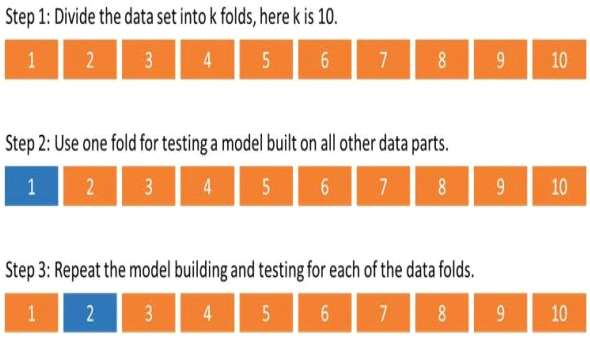

Figure 11: *Fold cross-validation [46]*

3.7.1 Related Work in SDN based DDoS attack detection

We discuss the related work based on two groups of detection methods. One group uses statistical approach while the other group uses machine learning approach.

3.7.2 DDoS attack Detection using Statistical Approach

Researchers can find many studies on DDoS attack detection methodologies. The method proposed by Seyed et al. [6] is based on the entropy comparison of consecutive packet samples to identify changes in their randomness. A window of 50 packets is collected, and the entropy is calculated from their destination IP addresses. If the entropy is less than the threshold, an attack is reported.

Surender Singh et al. [9] proposed a distributed framework, which analyzes the behavior of the packet flows. The proposed method uses entropy and a traceback algorithm to distinguish the malicious flows from the legitimate flow.

Jisa David et al. [10] proposed a DDoS attack detection system which is based on fast entropy using flow-based analysis. Their proposed method shows better detection accuracy. They analyze network traffic and compute the fast entropy of request per flow.

SPHINX [11] is a framework proposed to detect attacks in SDN in real-time with low performance overheads. It can detect both known and potentially unknown attacks on network topology. It is mainly based on an approximation of real network into a flow graph. It uses these Flow graphs to detect security threats in the network topology.

Lei et al. [12] proposed a system called FloodGuard. It concentrates on an SDN-specific attack called data-to-control plane saturation attack. It implements two modules proactive flow rule analyzer which preserves network policy enforcement and packet migration protects the controller being overloaded.

Qin et al. [13] proposed a method for intrusion detection with a time window of 0.1 seconds and three levels of threshold. This method tries to reduce false positive and false negative values. It is found that the time and resource consumption of the method is high.

Proposed method extends the recent work done by Kia [1] on Early Detection and Mitigation of DDoS Attacks in Software Defined Networks, which is based on an Entropy variation of the destination IP address, Flow Initiation Rate and Study of Flow Specification. The proposed method is a lightweight DDoS attack detection at its early stage. In our modified method we have implemented moving mean and standard deviation for the computation of adaptive thresholds and a better mitigation module has been introduced.

3.7.3 DDoS Attack Detection Using Machine Learning Approach

DDoS Attack Detection and Prevention based on Ensemble Classifier (RF) proposed by Alpna et al. [14] uses a combination of classifiers to improve the performance of their model. Experimental results were conducted on UCLA dataset. The results show high accuracy with minimum error.

In [15] [16], the authors have proposed network IDS using Random Forest algorithm. They have classified DDoS attack under network intrusion attacks. They have not considered the enormous volume of attack packets that DDoS detection system has to

handle in comparison with intrusion attacks. The method is only suitable to fight against the intrusion attacks. Moreover, their approach cannot be used to mitigate the attacks. Whereas in our thesis, we have classified the attack based on the features like payload size, packet count per flow and the flow duration that are responsible for the breakdown of the server. Our approach can also successfully mitigate the attacks.

Keisuke Kato et al. [17] proposed an intelligent DDoS attack detection system using packet analysis and Support Vector Machine. The detection system used SVM with a Radio Basis Function (RBF) neural networks. Experiments were done using CAIDA DDoS attack 2007 dataset.

Sivatha Sindhu et al. [18] proposed a neural decision tree for feature selection and classification. The proposed method uses six decision tree classifiers namely Decision Stump, C4.5, Naive Baye's Tree, Random Forest, Random Tree and Representative Tree model to detect the anomalous network pattern. They used sensitivity and specificity for the performance evaluation.

Saurav Nanda et al. [19] studied the attack patterns in the network using ML approach. The methodology uses 4 different ML algorithms like C4.5, Naive Bayes, Bayes net & Decision table. The prediction accuracy of the algorithms was compared, and they conclude that Bayesian network has the highest prediction rate.

Zhong et al. [20] proposed a DDoS attack detection method using a fuzzy c-means (FCM) clustering algorithm. To extract the features in network traffic they used Apriori association algorithm.

IDS using RF and SVM proposed by MD Al Mehedi Hasan et al. [21] developed 2 models for IDS using SVM and RF. The performance of these two models was compared based on their detection rate and precision and false negative rate.

The practical drawback in the above-mentioned approaches is that the authors have implemented the RF algorithm using the default prediction method which is the majority voting. Majority voting does not give accurate results out of random predictions. This drawback is eliminated in our approach by replacing majority voting by weighted voting, which gives more accurate results.

3.8 Mitigation Module

The next goal is to protect the switches and controller under attack. Usually, the controller will not crash easily as they are designed with high capacities. But the switches are not very robust against attacks due to their limited resources. During the attacks, the flow tables of the switches get filled with a large number of short flows which eventually breaks the switch.

Once the attack is detected, to prevent the breakdown of the switch, the default value of flow idle_timer is changed to the mitigating value. The mitigated value which is smaller than the default value makes the short flows timeout quickly and the flows are deleted from the switch flow tables.

4. METHODOLOGY

4.1 Research Objective and Contribution

The main goal of this research is to develop a detection system to identify Distributed Denial of Service (DDoS) attacks in the SDN environment. In this thesis, we use the traffic parameters of normal traffic, such as payload size and packet per flow, to identify the attack. In the statistic approach, the means and standard deviations of these parameters are measured to compute various thresholds. These thresholds are used to distinguish the normal and attack traffic. Whereas in the machine learning approach, an RF classifier with appropriate modifications is implemented and used to classify the traffic.

Furthermore, a mitigation method is also proposed to mitigate the effect of the attack.

4.2. UCLA Dataset

Building the training data is the most important step in the implementation of machine learning approach. We need to select adequate dataset for training our machine learning model. In this research, we have used UCLA dataset [22] to build the training data. The UCLA dataset contains real-time UDP flood attack traces. As our thesis is based on SDN architecture, we chose to modify the UCLA dataset by adding traffic flow entries of the simulated traffic in addition to the real traces. The data from the dataset is downloaded and preprocessed for any missing values and then converted to a comma separated file (.csv) format which can be read by the machine learning module developed in python. The features in UCLA dataset include Packet_TIME, IP_from, IP_to, PORT_from, PORT_to, LENGTH.

4.2.1 Features in UCLA Dataset

	A	B	C	D	E	F	G	H
1	Duration	From IP	To Ip	tp_src	tp_dst	nw_proto	byte lengt	result
2	0.015675	1.1.139.1C	1.1.236.8	9998	2	17	1001	1
3	0.015674	1.1.139.1E	1.1.236.8	9997	3	17	1001	1
4	0.01574	1.1.139.92	1.1.236.8	9996	4	17	1001	1
5	0.015824	1.1.139.21	1.1.236.8	9995	5	17	1001	1
6	0.015945	1.1.139.71	1.1.236.8	9994	6	17	1001	1
7	0.016028	1.1.139.14	1.1.236.8	9993	7	17	1001	1
8	0.01612	1.1.139.13	1.1.236.8	9992	8	17	1001	1
9	0.0162	1.1.139.84	1.1.236.8	9991	9	17	1001	1
10	0.016283	1.1.139.8C	1.1.236.8	9990	10	17	1001	1
11	0.016373	1.1.139.21	1.1.236.8	9989	11	17	1001	1
12	0.016457	1.1.139.8C	1.1.236.8	9988	12	17	1001	1
13	0.01654	1.1.139.61	1.1.236.8	9987	13	17	1001	1
14	0.016621	1.1.139.91	1.1.236.8	9986	14	17	1001	1
15	0.016691	1.1.139.23	1.1.236.8	9985	15	17	1001	1
16	0.016766	1.1.139.15	1.1.236.8	9984	16	17	1001	1
17	0.016857	1.1.139.17	1.1.236.8	9983	17	17	1001	1
18	0.01694	1.1.139.69	1.1.236.8	9982	18	17	1001	1
19	0.017028	1.1.139.1	1.1.236.8	9981	19	17	1001	1
20	0.017114	1.1.139.1C	1.1.236.8	9980	20	17	1001	1
21	0.0172	1.1.139.9C	1.1.236.8	9979	21	17	1001	1
22	0.01728	1.1.139.12	1.1.236.8	9978	22	17	1001	1
23	0.01737	1.1.139.83	1.1.236.8	9977	23	17	1001	1
24	0.017453	1.1.139.25	1.1.236.8	9976	24	17	1001	1
25	0.017541	1.1.139.15	1.1.236.8	9975	25	17	1001	1
26	0.017627	1.1.139.2C	1.1.236.8	9974	26	17	1001	1

Figure 12: *Sample UCLA Dataset*

Table 1: Features in UCLA Dataset

Features in UCLA Dataset	
Packet_TIME	Time when the packet
IP_fro	Number masking the IP
IP_to	Number masking the IP address of the packet destination
PORT_from	Original source port
PORT_to	Original destination port
U	UDP Packet
LENGTH	Length of packet (without header) in Bytes

1. Design and implementation of a DDoS attack detection system based on the statistical approach proposed by Kia [1]. We have modified and improved the approach of [1] by computing the threshold values based on the mean and standard deviations of the normal traffic parameters.
2. Propose an efficient mitigation method based on pushing a drop flow to block the attack traffic, thus, protect the controller and switch.
3. Design and implementation of a DDoS attack detection system based on the machine learning model using the Random Forest algorithm. The RF algorithm is modified in such a way that it uses weighted voting instead of standard majority voting for attack prediction as used by Alphna et al [14], Malik et al [15], Farnaaz et al [16].
4. Compare, analyze and evaluate the proposed detection and mitigation techniques with the approaches found in the literature.

4.3 Training Phase

With a training dataset and features selected, we can now train the machine learning models. The training phase is shown in Figure 12

The first step is to get the input dataset (UCLA), then process it. That is, the features or columns that have zero values needs to be edited or removed depending on the importance of the data. Also, the values containing characters must be converted into numeric in order for the data to be processed by the algorithms. The next step is to select the features which are relevant to the attack detection. We then train the models using random forest algorithm from python scikit-learn libraries, and finally, we save the trained model and record the results of cross-validation and use them for future predictions.

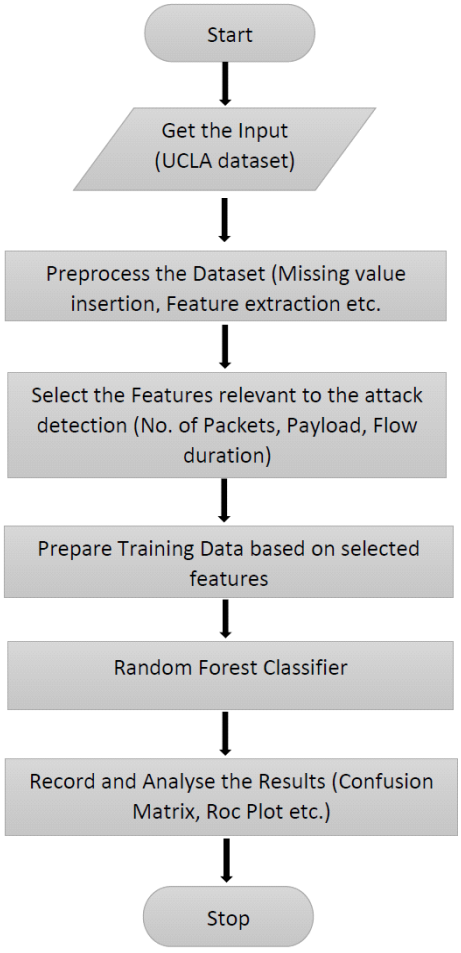

Figure 13: Flowchart for Training Phase

4.4 Testing Phase

Figure 13 shows the testing phase in the machine learning model of our proposed system. Get the input from the controller every 10 seconds and pass it as the input to the random forest classifier model built in training phase. Check if attacked is detected. If the attack is detected raise the alarm and record the attack to a log file. Otherwise, continue the process of getting the input.

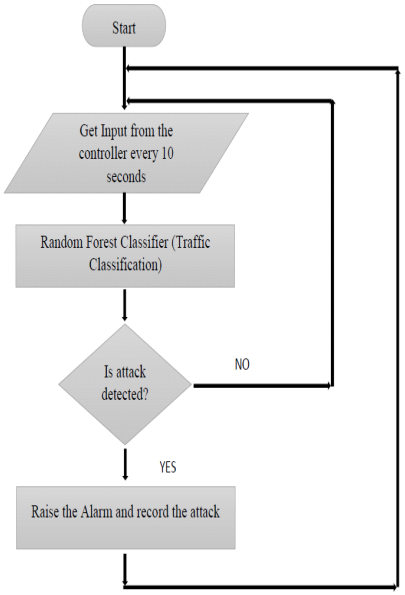

Figure 14: Flowchart for Testing Phase

4.5 Preparation of Training Data

The first step in our proposed detection method is the preparation of training data. The dataset downloaded from UCLA website is converted to a .csv file and the duplicate values are removed, and the missing parameters are added. After processing the dataset manually, it is loaded by the detection script by using the read_csv("filename") function. The .csv file is converted to pandas DataFrame to perform the classification process the DataFrame consists of two types of data: feature data labeled as X and target data labeled as Y. Every row in the X is a datapoint (i.e. a network traffic flow) and every column in X are a feature (e.g. byte length, packet count). For a classification problem, Y contains the class value (attack or no attack) of every data point. The Y column of our DataFrame is removed and saved as a

separate numpy array (python data structure) labeled as the Result. Now the remaining DataFrame has only the feature data labeled X. The feature information from the DataFrame is saved as a numpy array (matrix X) using the pandas function as_matrix.

The Result array with class value: attack or no attack is replaced with binary values 1 and 0. The 1 represents attack and 0 represents no attack

4.6 Training the classifier:

Once the training data is ready, we build an RF classifier and apply the classifier to the training data and use 10-fold cross-validation to compute the accuracy of the classifier. The RF classifier in our proposed method is built using scikit-learn python library. The next step is to generate decision trees. The number of decision trees generated can affect your accuracy. With the increase in the number of decision trees the accuracy also increases. The number of decision trees generated can be specified. In our thesis, we have generated 100 decision trees to provide better detection accuracy without introducing significant processing overhead.

By default, random forest decision trees are generated out of random samples from the training data. Also, splitting on a feature in the decision tree is done by considering a random subset of variables to split on. This randomness may affect the prediction accuracy.

4.7 Traffic generator

Our research is based on UDP flood attack and we use the traffic generator Scapy to generate UDP packets and spoof the source IP address of the packets. Scapy is a powerful interactive packet manipulation program written in python. It can forge packets of different protocols. It can perform tasks like scanning, tracerouting, probing, unit tests, attacks, and network discovery. It is used as a replacement of Hping, Arpspoof, Arping, TCPDUMP, etc.

In this I presented the code for generating both the normal and attack traffic. Figure 15 shows the traffic generated using Scapy script.

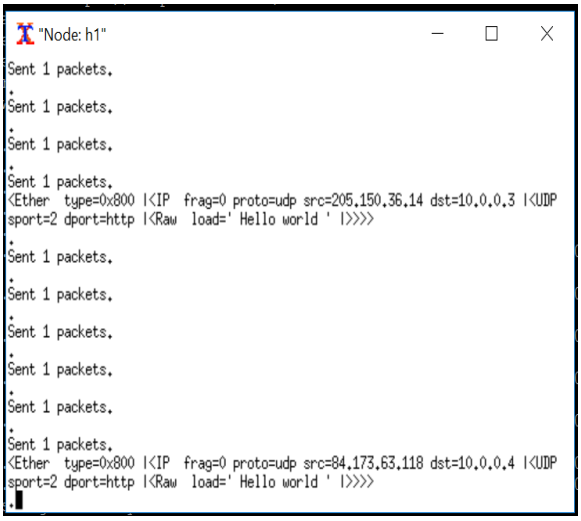

Figure 15: Traffic Generation using Scapy

The normal traffic patterns and attack traffic pattern used in the project are given below in table 2 and 3 respectively.

Table 2: Normal Traffic Pattern

Normal Traffic Pattern	
Packet Type	UDP
Packet Payload	60 bytes
No. of Packet Sent per Flow	Random between 1 and 8
Packet Inter Arrival Interval	0.1 Sec
Traffic Rate	10 Packets/Sec

Table 3: Attack Traffic Pattern

Normal Traffic Pattern	
Packet Type	UDP
Packet Payload	60 bytes
No. of Packet Sent per Flow	Random between 1 and 8
Packet Inter Arrival Interval	0.1 Sec
Traffic Rate	10 Packets/Sec

5. RESULTS AND DISCUSSION

5.1 Implementation of RF Classifier in the Controller

Every 10 seconds, the pox controller sends the flow statistics collected from the switches to the RF classifier. The model accepts these as unseen inputs and tries to predict if there is an attack.

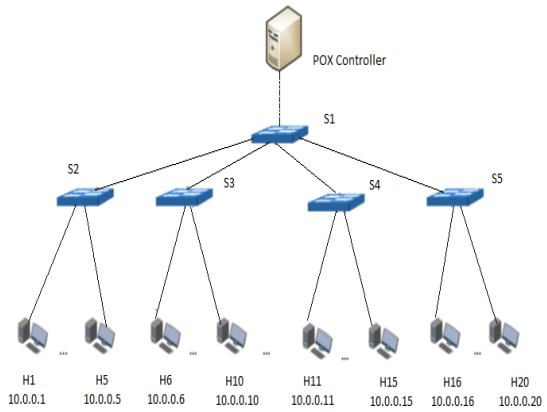

Figure 16: Network Setup

In the random forest method, the prediction is done by calling the predict_proba method. Once the method is called, it returns the prediction probabilities. For example: at a given point X there is a 60% probability that it belongs to class 1 and 40% probability that it belongs to class 0. The classifier's probabilities are converted to predictions. We can visualize the predicted probabilities using the predict_proba method.

However, when the classes in the training data are unbalanced (e.g. when the number of attack class is more compared to the no attack class), the predictions calculated by the classifier become inaccurate. This happens because our RF classifier learns the pattern of the training data to predict the unseen output. When the training data itself is unbalanced, the results turn out to be inaccurate.

The default behavior of random forest can be changed by choosing an appropriate threshold value. The analysis of precision rate can be helpful in choosing the appropriate threshold probability.

In our thesis, the value of output probability threshold is tuned so that we get the higher precision rate. More specifically, we have changed this default threshold α to 0.25 based on the analysis of the precision rate of the training set. The precision rate is calculated based on True Positive rate (TP) and False Positive rate (FP):

Precision = TP/(TP+FP)

The definitions of TP and FP are defined in the subsequent chapter. The implementation of weighted voting instead of majority voting improves the precision

5.2 Simulation Scenario and Results of Our Proposed Method

The experiment was done on a DELL Inspiron 5558 laptop with Intel (R) Core (TM) i5-5250U CPU @1. 60GHz, 1601 MHz, 2 Core(s), 4 Logical Processor(s). Using Mininet, a tree-type network is created. Its depth is two with five switches and 20 hosts. Figure 16 shows the network. Open Virtual Switch (OVS) was used for network switches. OVS is a software switch that runs both on hardware and software. In Figure 16, all switches refer to OpenFlow-enabled switches. The L2_multi.py module of POX was used for the controller.

The experiment mainly concentrates on multiple victim attacks as the single victim attack can be easily detected. During the simulation, we are running two Scapy programs. The one that is generating the attack sends the packets faster than the one which generates normal traffic. The traffic pattern shown in Table 4.1 and Table 4.2 is used for this purpose. We run attack traffic from randomly chosen 4 hosts to attack four different destinations, consequently remaining 16 hosts generate the legitimate traffic. The IP address in Mininet for all hosts are assigned from 10.0.0.1 to 10.0.0.20.

5.3 Performance of the Statistical Approach:

Based on the calculated threshold values, the performance of our detection system is analyzed by running the simulation 50 times. For each run, attack traffic of pattern B is generated on four hosts. Each simulation lasts about 30 minutes.

6. CONCLUSION

Detecting and defending against DDoS attack is a complex task. Initially, we started the research aiming to improve the DDoS detection system proposed by Kia [1]. We used the mean and standard deviation to compute various thresholds. We also introduced a better mitigation method to protect the controller and the switches. Even though we managed to improve the detection rate performance, we found that it is still not entirely satisfactory. Hence, to improve the detection rate further, we proposed an ML approach. Our proposed approach is based on RF algorithm with weighted voting. Our results show that the proposed approach has the best performance among all the approaches considered in this thesis.

7. FUTURE WORK

As a future work, we recommend developing a controller that can detect any type of network attack and employ deep packet inspection so that the detection accuracy is even higher. Moreover, SDN has also a concept of distributed network security enforcement by making each network element potentially an enforcement node and smart. This can be easily achieved by Machine Learning approach.

REFERENCES:

[1] Maryam Kia, "Early Detection and Mitigation of DDoS Attacks in Software Defined Networks", Master's thesis, Ryerson University, Canada 2015.

[2] Javed Ashraf and Seemab Latif, "Handling Intrusion and DDoS Attacks in Software Defined Networks Using Machine Learning Techniques", National Software Engineering Conference, 2014.

[3] opennetworking.org, "OpenFlow", [Online]. Available: https:// www.opennetworking.org. [Accessed February 2016].

[4] slideshare.net, "OpenFlow Tutorial", [Online] Available: https:// slideshare.net/openflow/.[Accessed September 2016].

[5] Sukhveer Kaur, Japinder Singh and Navtej Singh Ghumman, "Network Programmability Using POX Controller", International Conference on Communication, Computing & Systems, pp. 134-138, 2014.

[6] Seyed Mohammad Mousavi and Marc St. Hilaire, "Early Detection of DDoS Attacks against SDN Controllers", International Conference on Computing, Networking and Communications, Communications and Information Security Symposium, 2015.

[7] Duohe Ma1, Zhen Xu, and Dongdai Lin, "Defending Blind DDoS Attack on SDN Based on Moving Target Defense", researchgate, November 2015.

[8] Meiling Liu, Xiangnan Liu, Jin Li and Jiale Jiang, "Evaluating total inorganic nitrogen in coastal waters through the fusion of multi-temporal RADARSAT-2 and optical imagery using random forest algorithm", International Journal of Applied Earth Observation and Geoinformation 33 (1), pp. 192-202, December 2014.

[9] Surender Singh and Sandeep Jain, "A Review of Detection of DDoS Attack Using Entropy- Based Approach", IJCST Vol 4, Issue 2, April 2013.

[10] Jisa David and Ciza Thomas, "DDoS Attack Detection using Fast Entropy Approach on Flow-Based Network Traffic", Procedia Computer Science, pp. 30 – 36, 2015

[11] Maryam Kia, "Early Detection and Mitigation of DDoS Attacks in Software Defined Networks", Master's thesis, Ryerson University, Canada 2015.

[12] Javed Ashraf and Seemab Latif, "Handling Intrusion and DDoS Attacks in Software Defined Networks Using Machine Learning Techniques", National Software Engineering Conference, 2014.

[13] opennetworking.org, "OpenFlow", [Online]. Available: https:// www.opennetworking.org. [Accessed February 2016].

[14] slideshare.net, "OpenFlow Tutorial", [Online] Available: https:// slideshare.net/openflow/.[Accessed September 2016].

[15] Sukhveer Kaur, Japinder Singh and Navtej Singh Ghumman, "Network Programmability Using POX Controller", International Conference on Communication, Computing & Systems, pp. 134-138, 2014.

[16] Seyed Mohammad Mousavi and Marc St. Hilaire, "Early Detection of DDoS Attacks against SDN Controllers", International Conference on Computing, Networking and Communications, Communications and Information Security Symposium, 2015.

[17] Duohe Ma1, Zhen Xu, and Dongdai Lin, "Defending Blind DDoS Attack on SDN Based on Moving Target Defense", researchgate, November 2015.

[18] Meiling Liu, Xiangnan Liu, Jin Li and Jiale Jiang, "Evaluating total inorganic nitrogen in coastal waters through the fusion of multi-temporal RADARSAT-2 and optical imagery using random forest algorithm", International Journal of Applied Earth Observation and Geoinformation 33 (1), pp. 192-202, December 2014.

[19] Surender Singh and Sandeep Jain, "A Review of Detection of DDoS Attack Using Entropy- Based Approach", IJCST Vol 4, Issue 2, April 2013.

[20] Jisa David and Ciza Thomas, "DDoS Attack Detection using Fast Entropy Approach on Flow-Based Network Traffic", Procedia Computer Science, pp. 30 – 36, 2015.

[21] lasr.cs.ucla.edu, "UCLA Dataset", [Online]. Available: https:// lasr.cs.ucla.edu/DDoS/traces/ [Accesssed April 2017].

[22] mininet.org, "Mininet", [Online]. Available: http://mininet.org/. [Accessed June 2016].

[23] secdev.org, "Scapy", [Online]. Available: http://www.secdev.org/. [Accessed January 2016].

[24] Xin Xu and Xuening Wang, "An adaptive network intrusion detection method based on PCA and support vector machines", Advanced Data Mining and Applications, Springer, pp. 696-703, 2005.

[25] Ming-Yang Su, "Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest -neighbor classifiers", Expert Systems with Applications, PP. 3492-3498, 2011.

[26] Sindia and Julia Punitha Malar Dhas, "SDN Based DDoS Attack Detection and Mitigation in Cloud", IJCTA, pp.

39-47, 2017.

[27] Kuan-yin Chen, Anudeep Reddy Junuthula, Ishant Kumar Siddhrau, Yang Xu, H. Jonathan Chao, "SDNShield: Towards More Comprehensive Defense against DDoS Attacks on SDN Control Plane", IEEE, 2016.

[28] Christos Douligeris and Aikaterini Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art", Computer Networks, pp. 643–666, 2004.

[29] Mohammed Alenezi and Martin J Reed, "Methodologies for detecting DoS/DDoS attacks against network servers", The Seventh International Conference on Systems and Networks Communications, 2012.

[30] Manjula Suresh and R. Anitha, "Evaluating Machine Learning Algorithms for Detecting DDoS Attacks", Communications in Computer and Information Science, January 2011.

[31] Stefan Seufert and Darragh O'Brien, "Machine Learning for Automatic Defence against Distributed Denial of Service Attacks", ICC, 2007.

[32] Bhatia, Sajal, Schmidt, Desmond, & Mohay, George M, "Ensemble based DDoS detection and mitigation model", Fifth International Conference on Security of Information and Networks, pp.79-86, 2012.

[33] Yang Xu and Yong Liu, "DDoS Attack Detection under SDN Context", IEEE INFOCOM, 2016.

[34] pythonprogramming.net, "Python for Machine Learning", [Online]. Available: https://pythonprogramming.net/machine-learning-tutorial-python-introduction [Accessed June 2017].