

Network Service attack detection in Software defined networking

A Project Report Submitted in partial fulfillment of the degree of
Master of Computer Applications

By
Samit Kumar Das (20MCMC12)
Sandeep Singh (20MCMC05)



School of Computer and Information Sciences
University of Hyderabad, Gachibowli Hyderabad
- 500046, India

JUNE, 2022



CERTIFICATE

This is to certify that the Project Report entitled ” **Network Service attack detection in Software defined networking**” sub- mitted by **Samit Kumar Das** bearing Reg. No. 20MCMC12 and **Sandeep Singh** bearing Reg. No. 20MCMC05 in partial fulfillment of the requirements for the award of Master of Computer Applications, is a bonafide work carried out by them under my/our supervision and guidance.

The Project Report has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Dr. M A Saifullah

School of Computer and Information Sciences, University
of Hyderabad

Dr. Sanjay Sharma

Director, CNF, University of Hyderabad

Dean,

School of Computer and Information Sciences, University
of Hyderabad

DECLARATION

We, **Samit Kumar Das** and **Sandeep Singh** hereby declare that this dissertation entitled “**Network Service attack detection in Software defined networking**” submitted by us under the guidance and supervision of **Dr. M A Saifullah** and **Dr. Sanjay Sharma** is a bonafide work. We also declare that it has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

Date:

Samit Kumar Das
Reg. No.: 20MCMC12

Sandeep Singh
Reg. No.: 20MCMC05

Signature of the Student

Signature of the Student

Acknowledgments

We would like to express our gratitude to all those who gave us the support to complete this project.

First and foremost, we would like to gratefully acknowledge the enthusiastic supervision of our guides, **Dr. M A Saifullah** and **Dr. Sanjay Sharma**. They always allowed us a complete freedom to define and explore my own directions in research. Without their precious suggestions this thesis would not appear in the present form. We are also thankful for providing us an excellent lab facility, where we could test my skills effectively.

We would like to extend my sincere thanks to the **Dean** of School of Computer and Information Sciences for his valuable cooperation.

Finally we express my gratitude to all my family members and friends for their love and affection, without which this work would not have been possible.

Samit Kumar Das

MCA

Sandeep Singh

MCA

Abstract

The purpose of the project entitled as “**Network Service attack detection in Software defined networking**” is to detect Distributed Denial of Service (DDoS) attacks in the data plane. Distributed Denial-of-Service (DDoS) attacks represent a persistent threat to modern telecommunications networks: detecting and counteracting them is still a crucial unresolved challenge for network operators.

Software Defined Networking (SDN) provides well-known advantages in terms of network automation, flexibility and resources utilization, it has been observed that SDN controllers may represent critical points of failure for the entire network infrastructure, especially when they are targeted by malicious cyber-attacks such as Distributed Denial of Service (DDoS).

To address this issue, in this project we exploit data planes, where switches maintain persistent memory of handled packets to perform attack detection directly at the data plane, with only marginal involvement of the SDN controllers.

As machine learning (ML) is recognized as primary anomaly detection methodology, we perform DDoS attack detection using a ML based classification and compare different ML algorithms in terms of classification accuracy and train/test duration.

Moreover, we combine ML and software-enabled data planes to design a real-time DDoS attack detection module, which we evaluate in terms of latency required for the detection.

Keywords: Machine learning, software-defined network, distributed denial of services, protection, artificial neural network, decision trees, naïve bayes, security

Contents

Acknowledgments	iii
Abstract	iv
1 Introduction	9
1.1 What is SDN?	10
1.1.1 How does SDN work?	11
1.1.2 SDN with OpenFlow.	11
1.2 What are DDoS attacks?.	12
1.2.1 How does DDoS attacks work?.	13
1.2.2 How to identify a DDoS attack?.	13
1.2.3 Common types of DDoS attacks.	14
1.2.3.1 ICMP Flood DDoS Attack.	14
1.2.3.2 SYN Flood DDoS Attack.	15
1.2.3.3 UDP Flood DDoS Attack.	16
1.3 Motivation to present work	16
1.4 Organization of the Project	17
2 Integrating Machine Learning with SDN	18
2.1 DDoS attack in SDN.	19
2.2 Material and Method.	20
2.2.1 Data set.	20
2.2.2 Feature Selection in Machine Learning.	22
2.2.2.1 Filter methods.	22
2.2.2.2 Wrapper Methods.	22
2.2.2.3 Embedded Methods.	22
2.2.3 Machine Learning Classifiers.	23
2.2.3.1 Random Forest.	23
2.2.3.2 Support vector Machines.	23
2.2.3.3 K-nearest neighbors.	23
2.2.3.4 Naïve Bayes algorithm.	24
2.2.3.5 Decision Trees.	24
3 DDoS Detection Model: Inside view	25
3.1 Experimental results.	25
4 Proposed Work	31
4.1 RYU Controller.	31
4.2 Mininet.	31
4.3 Detecting DDoS attacks in RYU and Mininet Using ML.	33
4.3.1 Creating Dataset.	33
4.3.2 Running ML algorithm on Dataset.	34

4.3.3	Detecting legitimate and DDoS traffic.	35
4.3.3.1	Detecting legitimate traffic.	36
4.3.3.2	Detecting DDoS traffic.	37
5	Conclusion and Future Work	41
5.1	Conclusion.	41
5.2	Future Work.	41
	Bibliography	42

List of Figures

1.1	Layers of SDN	10
1.2	OpenFlow Architecture.....	12
2.1	Systematic diagram of attack in SDN controller.....	19
2.2	Systematic diagram for the implementation of features selection and Machine learning method.....	20
3.1	Machine Learning Based DDoS detection Model.....	25
3.2	Accuracy curve for information gain.....	26
3.3	Accuracy curve for correlation coefficient.....	26
3.4	Accuracy curve for chi-square.....	27
3.5	Accuracy curve for FFS.....	27
3.6	Accuracy curve for BFE.....	28
3.7	Accuracy curve for RFE.....	28
3.8	Accuracy curve for Lasso.....	29
3.9	a) Classifier result in term of precision, b) Classifier result in terms of Recall ,c) Classifiers result in term of specificity.....	30
3.10	Performance comparision of the proposed method with existing solutions in term of accuracy.....	30
4.1	Generating Normal traffic.....	33
4.2	Collecting Normal traffic.....	34
4.3	Generating DDoS traffic.....	34
4.4	Collecting DDoS traffic.....	34
4.5	Running ML algorithm on Dataset.....	35
4.6	Executing controller on RYU.....	35
4.7	Network Topology.....	36
4.8	Using ICM ping to generate normal traffic.....	36
4.9	Legitimate traffic detected.....	36
4.10	Performing syn attack.....	37
4.11	DDoS traffic on host 1 shown on controller window.....	38
4.12	Performing UDP Flood.....	38
4.13	Detecting UDP Flood.....	39
4.14	Performing ICMP Flood.....	39
4.15	Detecting ICMP Flood.....	39
4.16	Performing Smurf attack.....	40
4.17	Detecting Smurf attack.....	40

List of Tables

Table-1 Features of dataset.....21

Table-2 Comparison of experimental results.....29

The software-defined network (SDN) paradigm has gained the most significant interest in current days. The data centers and operator networks are shifting from traditional networks to SDN based networks because, it provided more reliable, flexible and secure network environment. Consequently, the deployment of the SDN in data centers and cloud computing environments provide reliable and flexible network architecture. The separation of control and data planes is the main innovation behind the SDN. Furthermore, the SDN provides an intelligent centralization that consists of controllers that manage the forward packet devices and the well-designed configuration like (Open-Flow) of these devices is essential. In the SDN, network devices like switches only forward logic, whereas the decision making and control logic ability are software at SDN controller.

The SDN controller makes the new network flow policies and instructs the switches to follow the new policies maintained in a flow table. Despite all these impressive innovations, the SDN-based network environment's several components pose some additional security threats to the SDN controller. So, the security of the SDN controller is becoming a key challenge for the development of future SDN base networks. As far as many issues to be addressed, all the security of the SDN controller is considered the highest concern. Distributed denial of service (DDoS) is a critical security issue for the SDN controller. The main aim of a DDoS attack is to make the computing resources unavailable for the users. The DDoS attack is usually caused by one or more than one bot, which is penetrated by software from malicious code. The DDoS attack also become a cause of massive damage to the network because it spreads more quickly.

Furthermore, in the SDN infrastructure, when the controller is under a DDoS attack, both forwarding and controller layers suffer from resource depletion. The existing research has shown improvement in detecting DDoS attacks at the control layer. A large dimensional and large volume of network traffic data is used in machine learning-based techniques. Selecting the most relevant features from the data set for the training and testing of machine learning models is still an open issue. The optimal selection of packet features is important because it influences the design of the efficient ML-based detection model for the DDoS in SDN. Motivated by this, in this paper, a comparative analysis of feature selection based ML classifiers is present for the efficient detection of DDoS attacks in the SDN. Different feature selection techniques that include filter, wrapper, and embedded are used to find out the most optimal subset of features for the training and testing of ML algorithms. The optimal selection of features subset is significant because it reflects the overall accuracy of the machine learning classifier. The literature survey related to the SDN has concluded that the SDN controller's control plane exhibit more vulnerabilities for the DDoS attack, and most of the researchers preferred ML techniques. Moreover, selecting the optimal features from the data set is still insignificant because the use of a large number of features for the ML models increases the cost and time complexities. Furthermore, the use of irrelevant features is not able to detect the attack more

efficiently.

The main contribution of the project is as follow:

- This work utilizes different feature selection techniques to find the most optimal features for the training and testing machine learning models. The machine learning model with optimal features gives effective results for detecting DDoS attacks in SDN controllers.
- The presented analysis combines features selection techniques with a machine learning model. Further, to reduce the training time of the machine learning models, the subset of optimal features is used.
- The presented analysis compares the results of machine learning classifiers on a different subset of features that are ranked by the feature selection methods. The experimental results have shown the effectiveness of the machine.

1.1. What is SDN ?

Software-Defined Networking (SDN) is an approach to networking that uses software-based controllers or application programming interfaces (APIs) to communicate with underlying hardware infrastructure and direct traffic on a network.

This model differs from that of traditional networks, which use dedicated hardware devices (i.e., routers and switches) to control network traffic. SDN can create and control a virtual network – or control a traditional hardware – via software.

While network virtualization allows organizations to segment different virtual networks within a single physical network, or to connect devices on different physical networks to create a single virtual network, software-defined networking enables a new way of controlling the routing of data packets through a centralized server.

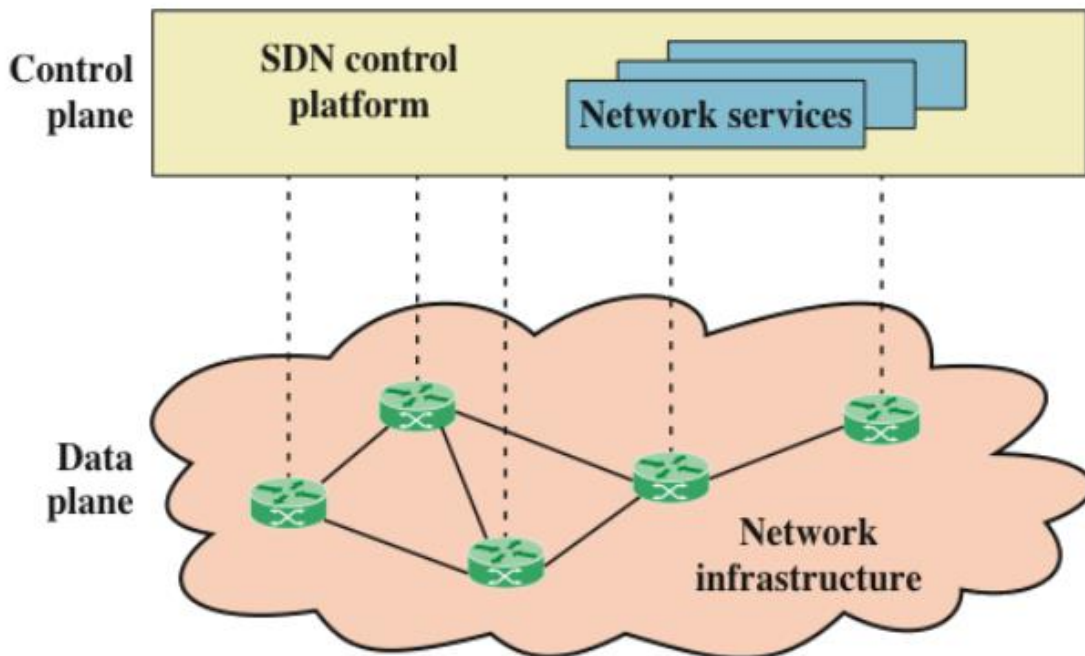


Fig 1.1: Layers of SDN

1.1.1. How does SDN work?

Here are the SDN basics: In SDN (like anything virtualized), the software is decoupled from the hardware. SDN moves the control plane that determines where to send traffic to software, and leaves the data plane that actually forwards the traffic in the hardware. This allows network administrators who use software-defined networking to program and control the entire network via a single pane of glass instead of on a device by device basis.

There are three parts to a typical SDN architecture, which may be located in different physical locations:

Applications, which communicate resource requests or information about the network as a whole.

Controllers, which use the information from applications to decide how to route a data packet.

Networking devices, which receive information from the controller about where to move the data

Physical or virtual networking devices actually move the data through the network. In some cases, virtual switches, which may be embedded in either the software or the hardware, take over the responsibilities of physical switches and consolidate their functions into a single, intelligent switch. The switch checks the integrity of both the data packets and their virtual machine destinations and moves the packets along.

1.1.2.SDN with OpenFlow

OpenFlow is a multivendor standard defined by the open networking foundation (ONF) for implementing SDN in networking equipment. The OpenFlow protocol defines the interface between an OpenFlow Controller and OpenFlow switch.

The OpenFlow switch may be programmed to:

- 1) Identify and categorize packets from an ingress port based on various packet header fields.
- 2) Process the packets in various ways, including modifying the header.
- 3) Drop or push the packets to a particular egress port or to the OpenFlow controller.

The OpenFlow instructions transmitted from an OpenFlow controller to an OpenFlow switch are structured as “flows”. Each individual flow contains packet match fields, flow priority, various counters, packet processing instructions, flow timeouts and a cookie. The flows are organized in tables. An incoming packet may be processed by flows in multiple “pipelined” tables before exiting

on an egress port.

The OpenFlow Network Architecture consists of three layers:

- 1) One or more Open Flow virtual and/or physical switches;
- 2) One or two OpenFlow controllers and
- 3) One or more OpenFlow applications.

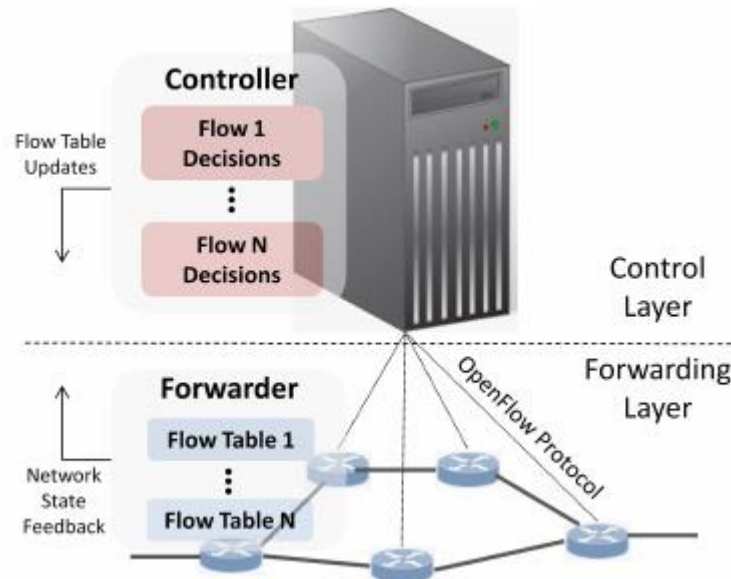


Fig 1.2: OpenFlow Architecture

The OpenFlow controller maintains the OpenFlow protocol communications channels to the OpenFlow switches, maintains a local state graph of the OpenFlow switches and exposes a northbound API to the OpenFlow applications. The northbound API may be viewed as an abstraction of the network and allows the OpenFlow applications to read the state of the network and to instruct the network to perform various tasks.

1.2. What are DDoS attacks?

A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of internet traffic.

DDoS attacks achieve effectiveness by utilizing multiple compromised computer systems as sources of attack traffic. Exploited machines can include computers and other networked resources such as IOT devices.

From a high level, a DDoS attack is like an unexpected traffic jam clogging up the highway, preventing regular traffic from arriving at its destination.

1.2.1. How does a DDoS attack work?

DDoS attacks are carried out with networks of Internet-connected machines.

These networks consists of computers and other devices (such as IoT devices) which have been infected with malware, allowing them to be controlled remotely by an attacker. These individual devices are reffered to as bots (or zombies), and a group of bots is called a botnet.

Once a botnet has been established, the attacked is able to direct an attack by sending remote instructions to each bot.

When a victim's server or network is targeted by the botnet, each bot sends requests to the target's IP address, potentially causing the server or network to become overwhelmed, resulting in a denial-of-service to normal traffic.

Because each bot is a legitimate Internet device, separating the attack traffic from normal traffic can be difficult.

1.2.2. How to identify a DDoS attack?

The most obvious symptom of a DDoS attack is a site or service suddenly becoming slow or unavailable. But since a number of causes — such a legitimate spike in traffic — can create similar performance issues, further investigation is usually required. Traffic analytics tools can help you spot some of these tell-tale signs of a DDoS attack:

- Suspicious amounts of traffic originating from a single IP address or IP range
- A flood of traffic from users who share a single behavioural profile, such as device type, geolocation, or web browser version
- An unexplained surge in requests to a single page or endpoint
- Odd traffic patterns such as spikes at odd hours of the day or patterns that appear to be unnatural (e.g. a spike every 10 minutes)

There are other, more specific signs of DDoS attack that can vary depending on the type of attack.

1.2.3. Common types of DDoS attacks:

1.2.3.1. ICMP Flood DDoS Attack:

An Internet Control Message Protocol (ICMP) flood DDoS attack, also known as a Ping flood attack, is a common Denial-of-Service (DoS) attack in which an attacker attempts to overwhelm a targeted device with ICMP echo-requests (pings). Normally, ICMP echo-request and echo-reply messages are used to ping a network device in order to diagnose the health and connectivity of the device and the connection between the sender and the device. By flooding the target with request packets, the network is forced to respond with an equal number of reply packets. This causes the target to become inaccessible to normal traffic.

Others types of ICMP request attacks may involve custom tools or code, such as hping and scapy. Attack traffic that emanates from multiple devices is considered DDoS attack. In this type of DDoS attack, both incoming and outgoing channels of the network are overwhelming, consuming significant bandwidth and resulting in a denial of service.

An ICMP flood DDoS attack requires that the attacker knows the IP address of the target. Attacks can be separated into three categories, determined by the target and how the IP address is resolved:

- Targeted local disclosed – In this type of DDoS attack, a ping flood targets a specific computer on a local network. In this case, the attacker must obtain the IP address of the destination beforehand.
- Router disclosed – Here, a ping flood targets routers with the objective of interrupting communications between computers on a network. In this type of DDoS attack, the attacker must have the internal IP address of a local router.
- Blind ping – This involves using an external program to reveal the IP address of the target computer or router before launching a DDoS attack.

Preventing an ICMP flood DDoS attack can be accomplished by disabling the ICMP functionality of the targeted router, computer or other device. By setting your perimeter firewall to block pings, you can effectively prevent attacks launched from outside your network. It's important to note that this approach won't prevent internal attacks. Also, when using IPv6, some ICMPv6 messages have to be permitted in order to maintain normal operations.

While eliminating the processing of the request and the Echo Reply will stop ICMP attacks, it will also make the device unresponsive to ping requests, traceroute requests, and other network activities, thus limiting the ability to diagnose server issues.

Another approach to combating ICMP attacks is to rate limit the processing of incoming ICMP messages; alternatively limit the allowed size of the ping requests.

1.2.3.2. SYN Flood DDoS Attack:

A SYN flood (half-open attack) is a type of DDoS attack which aims to make a server unavailable to legitimate traffic by consuming all available server resources. By repeatedly sending initial connection request (SYN) packets, the attacker is able to overwhelm all available ports on a targeted server machine, causing the targeted device to respond to legitimate traffic sluggishly or not at all.

SYN flood attacks work by exploiting the handshake process of a TCP connection. Under normal conditions, TCP connection exhibits three distinct processes in order to make a connection.

1. First, the client sends a SYN packet to the server in order to initiate the connection.
2. The server then responds to that initial packet with a SYN/ACK packet, in order to acknowledge the communication.
3. Finally, the client returns an ACK packet to acknowledge the receipt of the packet from the server. After completing this sequence of packet sending and receiving, the TCP connection is open and able to send and receive data.

To create dos attack, an attacker exploits the fact that after an initial SYN packet has been received, the server will respond back with one or more SYN/ACK packets and wait for the final step in the handshake. Here's how it works:

1. The attacker sends a high volume of SYN packets to the targeted server, often with spoofed IP addresses.
2. The server then responds to each one of the connection requests and leaves an open port ready to receive the response.
3. While the server waits for the final ACK packet, which never arrives, the attacker continues to send more SYN packets. The arrival of each new SYN packet causes the server to temporarily maintain a new open port connection for a certain length of time, and once all the available ports have been utilized the server is unable to function normally.

1.2.3.3.UDP Flood DDoS Attack:

A UDP flood is a type of denial-of-service attack in which a large number of User Datagram packets are sent to a targeted server with the aim of overwhelming that device's ability to process and respond. The firewall protecting the targeted server can also become exhausted as a result of UDP flooding, resulting in a denial-of-service to legitimate traffic.

A UDP flood works primarily by exploiting the steps that a server takes when it responds to a UDP packet sent to one of its ports. Under normal conditions, when a server receives a UDP packet at a particular port, it goes through two steps in response:

1. The server first checks to see if any programs are running which are presently listening for requests at the specified port.
2. If no programs are receiving packets at that port, the server responds with a ICMP (ping) packet to inform the sender that the destination was unreachable.

Most operating systems limit the response rate of ICMP packets in part to disrupt DDoS attacks that require ICMP response. One drawback of this type of mitigation is that during an attack legitimate packets may also be filtered in the process. If the UDP flood has a volume high enough to saturate the state table of the targeted server's firewall, any mitigation that occurs at the server level will be insufficient as the bottleneck will occur upstream from the targeted device.

1.2.3.4.Smurf Attack:

A Smurf attack is a form of a DDoS attack that renders computer networks inoperable. The Smurf program accomplishes this by exploiting vulnerabilities of the Internet Protocol (IP) and Internet Control Message Protocols (ICMP).

The steps in a Smurf attack are as follows:

- First, the malware creates a network packet attached to a false IP address — a technique known as "spoofing."
- Inside the packet is an ICMP ping message, asking network nodes that receive the packet to send back a reply
- These replies, or "echoes," are then sent back to network IP addresses again, setting up an infinite loop.

1.3. Motivation of the Present work

Now a days , even if internet technologies have advanced so much in terms of cyber security, information safety and threat detection still , DDoS attacks posed a big danger to all existing networking system environments. The main reason is that it is very hard to detect a DDoS attack because most of the time the incoming connection request seems like a normal request which , is not even detected by modern firewalls.

Software Defined Networking (SDN) can be used to propose a solution for this problem. By separating the control plane and the forwarding plane we can incorporate different methodologies in the software defined control plane to better facilitate threat detection and error

control. Using machine learning algorithms like SVM, Decision tree, K nearest neighbours etc. we can detect various kinds of DDoS attacks and take necessary action to stop the attack.

1.4. Organization of the Project

Our dissertation is organized as follows:

Chapter 2 presents how we have integrated machine learning with SDN; Chapter 3 presents the insight of the inner workings of our machine learning algorithms; Chapter 4 concludes and suggests possible future work.

Chapter 2

Integrating Machine Learning with SDN

Machine learning techniques give a compelling performance for the detection of DDoS attacks in SDN. The ML techniques effectively detect the attack against the control plane of the SDN controller. This section briefly discusses the current works to detect DDoS attacks in SDN using machine learning techniques. Furthermore, features selection based ML models and techniques presented by the researchers in recent years are analyzed in the section. In , a statistical and machine learning-based method is proposed. K-mean and K nearest neighbors (KNN) based hybrid model is proposed in. Support vector machine (SVM) based DDoS detection in SDN was performed in. Kernel principal component analysis (KPCA), Genetic algorithm (GA) and SVM based method is presented.

In an entropy-based technique that uses Flow samples are presented for traffic classification, and it just focuses on a standard distribution of the traffic. A COFFEE model that extracts the features from the flow for the detection of attack is present in. For the extraction of more features, the suspected flow sends to the controller. In many features are utilized by the machine learning techniques to detect the attack.

Furthermore, traffic features based on a lightweight DDoS attack detection algorithm is presented. The Self-organizing map (SOM) is used for the analysis and extraction of traffic information. After the extraction of features Artificial Neural Network is used for the detection of DDoS attacks. The researchers proposed a k-nearest neighbor-based algorithm that uses the abstract distance between the traffic features to detect the attack. This algorithm gives effective results for the detection of abnormal flow and also reduce the false alarm rate. Although the researchers proposed various machine learning-based solutions for detecting DDoS attacks, these solutions have some limitations in terms of optimal feature selection, low accuracy and efficiency.

Naïve bayes (NB) and K-mean clustering-based method was proposed to detect DDoS attacks. The K-mean cluster method clusters the traffic data that show similar behaviors and the Naïve Bayes algorithm classifies the cluster data into standard and attacks traffic. Artificial Neural Network-based methods are proposed to detect known and unknown DDoS attacks. The researcher in the controller applies a dynamic Multilayer perceptron (MLP) that works with a feedback mechanism to detect DDoS attacks. They use some selected features that cannot distinguish between standard and attack traffic flows.

The authors introduced the trigger mechanism that detects the DDoS attack faster and reduces the switches' workload. The trigger mechanism applied on the controller's control plane effectively attacks but increases the controller's workload.

2.1. DDoS attack in SDN

In a DDoS attack, the rate of incoming packets towards the network becomes high. Hence, the collection of spoofed and legitimate packets binds the network resources that make the resources unavailable. If this process continues, the server starts to drop the packets, and it becomes unreachable for the new incoming legitimate packets. The DDoS attack is categorized into three types: application-layer attack, protocol-exploitation attack, and volumetric attack. The TCP flooding and UDP flooding attacks are considered volumetric attacks, whereas the DNS and HTTP flood are referenced as application-layer attacks. The control plane of the SDN controller has centralized network intelligence. In a single SDN controller-based architecture, there is a high chance of a Single point of failure (SPF).

When the attacker gets access to the controller, it gives massive damage to the network's infrastructure. At the top of the control plane, the applications that include routing, firewall and load balancing are operated. If the attacker accesses the firewall application, the controller forms a different Access control list (ACL). A secure connection is created between the OF switch and controller using TLS/SSL; if the TLS connection goes on downstate, it needs a backup controller for the switch. In such a case, the OF switch will use the flow table according to his choices. A DDoS attack may create onto the controller when a malicious flow can be rule into the flow table. Besides this, in SDN, the format of the flow has some essential properties. The SDN controller uses the southbound protocol that includes OpenFlow to take action against the flow entries. The same flow in SDN maybe has more than one rule for it. The flow has different fields that include timeout, counter, action field, priority, etc. Each field has its specific task. For example, the timeout field represents the expiration time for a flow. The counter field keeps the information regarding bytes per flow, and the instruction field identifies the needed action for a flow entry. Fig. 2.1 describes the discussed scenario.

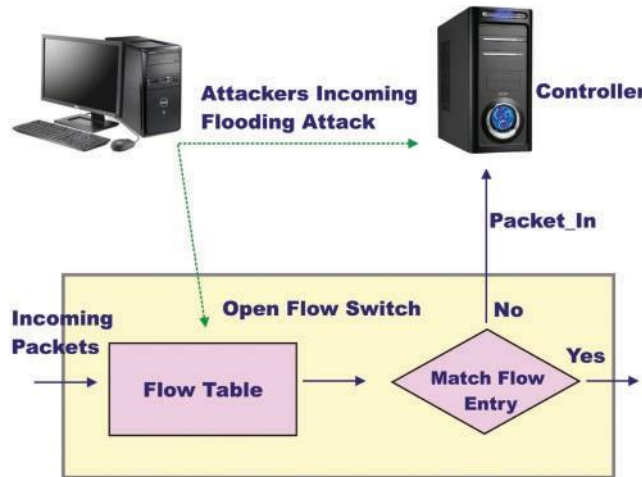


Fig 2.1: Systematic diagram of attack in SDN controller

2.2. Material and Method

This section describes the material and method of the paper. The systematic diagram of the process steps for the implementation of feature selection methods and machine

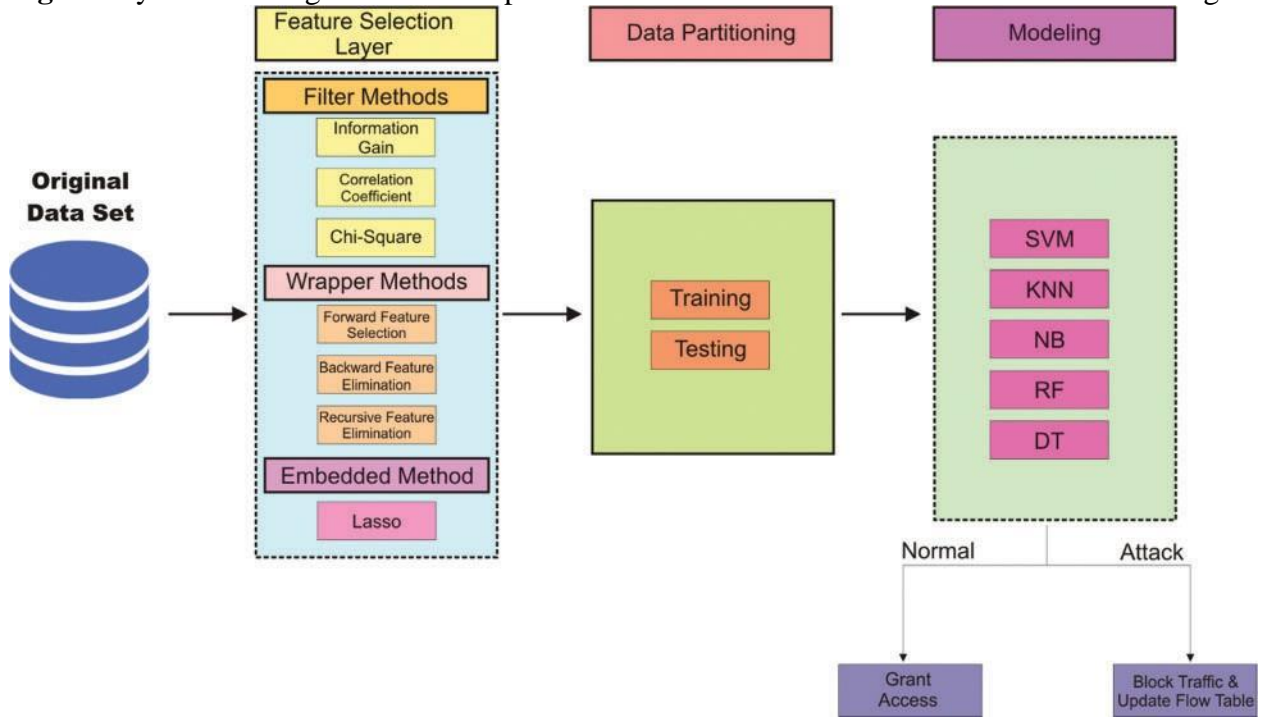
learning classifiers is presented in Fig 2.2. Additionally, this section also describes the description about dataset, feature selection methods and machine learning classifiers. This process consists of different phases that includes data selection, feature selection, data partitioning and modeling.

2.2.1. Data set

This section briefly describes the description of the data set that has been used in this work. NSL-KDD dataset is used to for the training and testing of machine learning classifiers for the detection of DDoS attack. The NSL-KDD is the refined version of the KDD'99 dataset. The data set has 41 features and 52,800 records that have been considered for the simulation.

The NSL-KDD has the data of different type of attacks that includes DDoS, Probe, R2L, U2R and normal traffic. The data related to DDoS attack is extracted from the original data set and it used to evaluate the performance of different machine learning classifiers for the detection of DDoS attack in SDN. The ML classifiers perform efficiently and evaluate accurately.

Fig 2.2: Systematic diagram for the implementation of features selection and machine learning



method.

Table 1: Features of the dataset

No.	Feature name	No.	Feature name
1	Duration	22	Srv-count
2	Source bytes	23	Serror-rate
3	Destination bytes	24	Srv-serror-rate
4	Land	25	Rerror-rate
5	Wrong fragment	26	Srv-rerror-rate
6	Urgent	27	Same-srv-rate
7	Hot	28	Diff-srv-rate
8	Number-of-failed-logins	29	Srv-diff-host-rate
9	Logged-in	30	Dst-host-count
10	Num – compromised	31	Dst-host-count
11	Root-shell	32	Dst-host-same-srv-rate
12	Su-attempted	33	Dst-host-diff-srv-rate
13	Num-rootNum-file-creations	34	Dst-host-same-src-port-rate
14	Num-shells	35	Dst-host-srv-diff-host-rate
15	Num-access-files	36	Dst-host-srv-diff-host-rate
16	Num-outbound-cmds	37	Dst-host-srv-serror-rate
17	Is-host-login	38	Dst-host-rerror-rate
18	Is-guest-login	39	Dst-host-srv-rerror-rate
19	Count	40	Protocol_type
20	Num_file_creations	41	Flag
21	Service		

2.2.2. Feature Selection in Machine Learning

In machine learning, feature selection task is important because the use of irrelevant features by the ML model can increase the running time and the cost of the system. It also

affects the performance of the models and makes the generalization performance of the models much poorer. In general form, the feature selection method for a learning problem from data can be formulated as: for a given set of labeled data samples $(x_1, y_1), \dots, (x_i, y_i)$ where $x_i \in R^n$ and $y_i \in R$ (or $y_i \in 1$ for a classification problem), the variable y_i achieved the lowest prediction error for a subset of m features ($m < n$). There are several algorithms that can be used for the selection of optimal features. In general, they can be structured into three different categories that includes filter, wrapper and embedded.

2.2.2.1.Filter methods

In filter methods, the variable ranking technique is used as a principle for the selection of features. Ranking methods are well known due to their simplicity and also have good success for practical applications. For the selection of the features, a suitable ranking criteria and threshold is used. The features that are below from the threshold value will be discarded from the feature subset. Furthermore, ranking methods, also known as filter methods, are applied on the dataset before classification to filter out less relevant features. The basic property of the optimal features is that they contain the information related to different classes that are present in the data. This property describes the relevance of the features and provides a way for the measurement of features usefulness for different classes. Furthermore, the relevant feature does not depend on input data, but they depend on the class label i.e., if the feature has no influence on the class labels than it can be discarded. The correlation is used in filter methods for the determinations of the unique or optimal features. The correlation and mutual information both methods are used in filter methods for the selection of optimal feature subset.

2.2.2.2.Wrapper Methods

The wrapper methods use itself as a regression or classification model and search the good subset of features through evaluation function. The classification model with different subsets of features is run on the training dataset. The features which produce lowest estimated error will be chosen as optimal features. The mathematical formulation of the wrapper methods as follows: The main goal of the feature selection method is to find the optimum n -column vector σ where $\sigma \in 0, 1$, that is defined as the optimal subset of selected features.

2.2.2.3.Embedded Method

The embedded feature selection methods are included in the classification and regression models. In these algorithms, the features that best contribute for the accuracy of the model will be selected as optimal subset of features. The wrapper based and filter based methods have been combined for the development of the embedded methods. Furthermore, they take the advantage of the feature selection process and performs the features selection and classification simultaneously. The Lasso embedded feature selection algorithm has been used for the selection of optimal features because it eliminates the weights of the least important features and gives a reduced set of features.

2.2.3. Machine Learning Classifiers

The set of optimal features which were selected by different feature selection methods are used as an input for different machine learning classifiers. The brief description of these classifiers is as below:

2.2.3.1. Random Forest

This section describes the general framework of the Random forest(RF) model. The RF classifier model consists of 1000 trees, and minimum number leaf node is 1. Furthermore, in RF model every weak learner was grown to its maximum, unpruned and 63% observations of the feature subset m was provided for the bootstrap, where m represents the number of features, and all optimal features are used by the RF model.

2.2.3.2. Support Vector Machines

The support vector machine model is implemented with radial basis kernel function where the optimal hyperplane is computed.

Subject to $y_i(w^T \phi(x_i) - b) \geq 1 - \phi_i$ where ϕ_i is $0, i = 1, \dots, N$, where C represents the cost factor that penalizes the miss-classification in the training data, w is the vector coefficient, b is the constant intercept term, and ϕ_i is used to control the margins on each side of hyperplanes, and ϕ represents the radial basis kernel function. In SVM, the radial basis kernel function gives better accuracy as compared to the linear kernel function.

2.2.3.3. K-nearest neighbors

K-nearest neighbors is a supervised machine learning classifier that is simple and can be easily used to solve regression and classification problems. The nearest k neighbors mechanism is used to determine the class for the new upcoming data. The Euclidean and Manhattan distance functions are used for the measurement of the distance between two data. In this paper, the Euclidean distance function is used. The similarity between the data samples that to be classified and the sample that were found in the classes was distinguished. The Euclidean distance function calculates the distance between the new encountered data and the data which is present in the training set individually. After that, the classification set is created by selecting the k dataset which has the smallest distance. The number of KNN neighbors is based on the value of classification.

2.2.3.4. Naïve bayes algorithm

The Naïve Bayes algorithm is based on the bayes rules that uses conditional independences of the features X , $[X_1, X_2, X_3, \dots, X_n]$ and corresponds to the output $Y \in [0, 1]$. Furthermore, the Gaussian NB model uses probability for the estimation class of continuous predictive features.

2.2.3.5. Decision Trees

The Decision tree (DT) consists of nodes and branches. It arranges the knowledge extracted from the data in the recursive hierarchical structure. Each internal node is represented as an attribute, and it is associated with the relevant data for classification. The classes in the data set are corresponding by the leaf nodes of the tree, and branches represent the possible results. The new input can be classified continuously by the nodes and branches until the leaf node is reached. The main aim of the induction process in the DT is to maximize the correct classification for all the training data.

Furthermore, the pruned process is applied to the trained tree to avoid overfitting. The decision tree also generates a comprehensive structure of the classification. For each new data, the final classification is determined through the verification of different attributes.

Chapter 3

DDoS Detection Model: Inside view

The machine learning-based model for the detection of DDoS attacks in SDN is presented in Fig. 3.1. The model monitors the OpenFlow (OF) switches for time intervals, and the SDN controller sends the flow results to all the switches present in the network. Furthermore, the SDN

controller receives the statistics of the flow, and after that, it fed these flows into the statistics monitor for the extraction of features. The extracted features send to the feature selection method to select the optimal features that are important for the detection of DDoS attacks. After feature selection, these features are sent to the machine learning classifiers that predict the normal and attack traffic flow. If the ML classifier detects the DDoS attack, then the mitigation module present in the SDN controller immediately sends new flow rules to the switches to drop the upcoming packets.

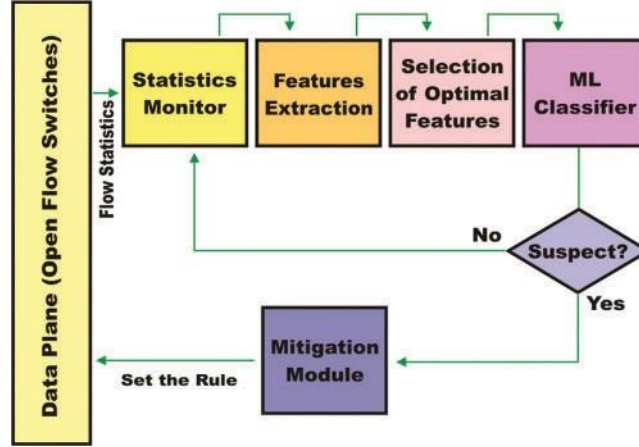


Fig 3.1 : Machine Learning Based DDoS detection Model

3.1. Experimental results

To evaluate the performance of the machine learning classifiers, different evaluation metrics that includes accuracy, precision and recall have been selected. Confusion matrix is used to compute these evaluation metrics. The performance evaluation of the machine learning classifiers is important for the accurately detection of attack in the SDN controller. The mathematical formulation of these metrics is as follow:

$$\text{Accuracy} = (TP + TN) / (TP + FP + TN + FN)$$

$$\text{Precision} = TP / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

$$\text{Specificity} = TN / (TN + FP)$$

Among these formulas, TP (True positive) is the probability of the attack traffic which is recognized as attack; TN (True negative) is the probability of normal traffic which is known as normal traffic; FP (False positive) is referred as the probability of the normal traffic which is recognized as attack traffic and FN (False negative) is the probability of the attack traffic which is recognized as normal traffic. The classifier which achieved the higher recall rate and accuracy gives better detection performance for the attack.

In this section, the performance of the various classifiers has been evaluated on different optimal features subsets that is selected by feature selection methods. The classifiers that include SVM, KNN, NB, RF and DT are selected. Furthermore, the filter, wrapper and embedded

methods are used to rank the most optimal features. These methods include Information Gain, Correlation Coefficient, Chi-Square, Forward Feature Selection, Backward Feature Selection, Recursive Feature Elimination and Lasso.

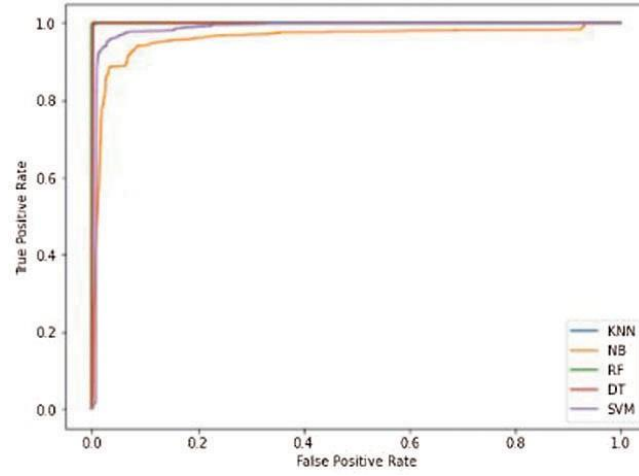


Fig 3.2: Accuracy curve for information gain

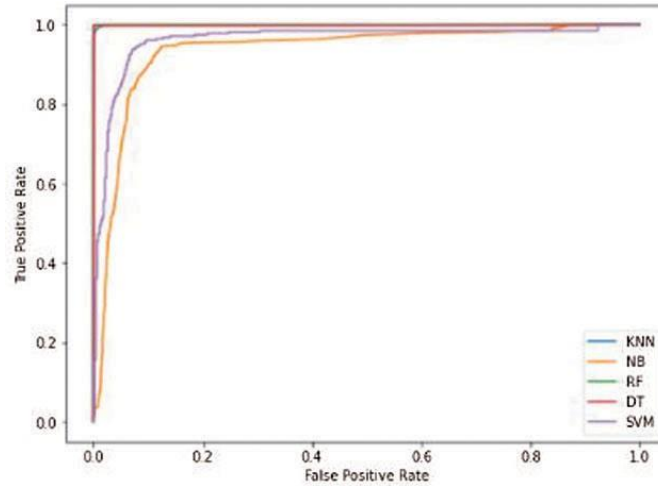


Fig 3.3: Accuracy curve for correlation coefficient

Based on the results, the classification accuracy of the different classifiers goes on the stable state on different subsets of the features. Here, stable accuracy is defined as some classifiers achieving maximum accuracy at the same set of features. Therefore, it is imperative to find the top-optimal features subsets on which classifiers offer better results for detecting the attack. Comparing the accuracy of the different classifiers, the RF classifiers consistently achieved 99.97% accuracy on the feature's subset, which is ranked by the RFE method. Therefore, our finding is that the RF classifier gives better results than the other classifiers for detecting DDoS attacks.

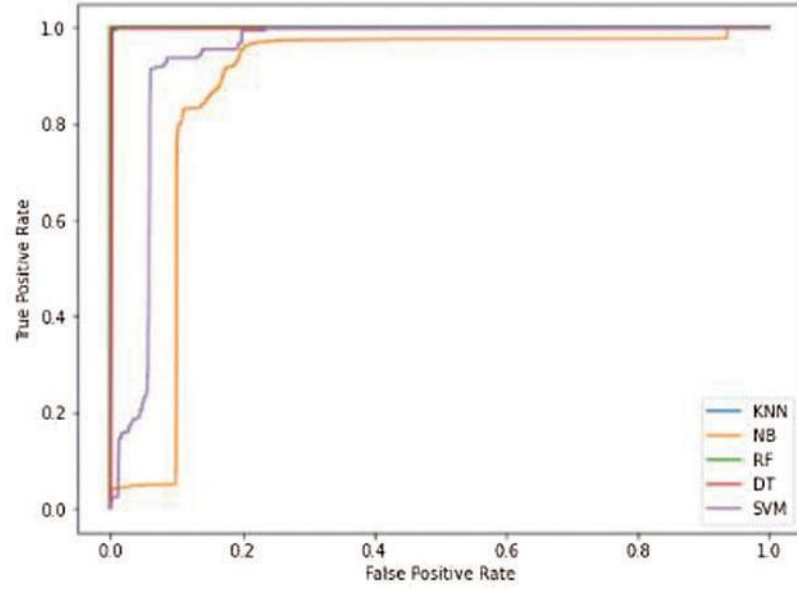


Fig 3.4: Accuracy curve for chi-square

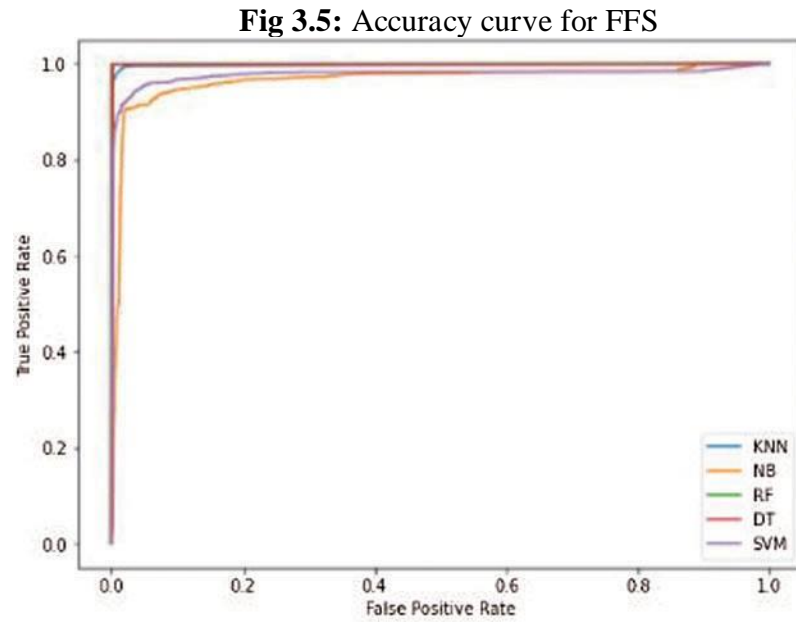


Fig 3.5: Accuracy curve for FFS

Furthermore, a comparative analysis of the different classifiers on different subsets of the optimal features is presented in Table. The table shows the number of features selected by each method and the accuracy of the classifiers on this subset of features. It is observed in the Table 3.1 that the maximum accuracy, which is 99.97%, is achieved by the RF classifier on the RFE feature subset. The favourable results of the RFE methods cloud are attributed to the fact that the RF classifier and RFE are both founded on similar information metrics. Furthermore, the other classifiers that include SVM, NB, KNN and DT also give better results on the RFE feature subset compared to the features subset that is ranked by the other feature selection methods. So, it is also

concluded that the 28 features selected by the RFE method are most optimal for detecting DDoS attacks in SDN.

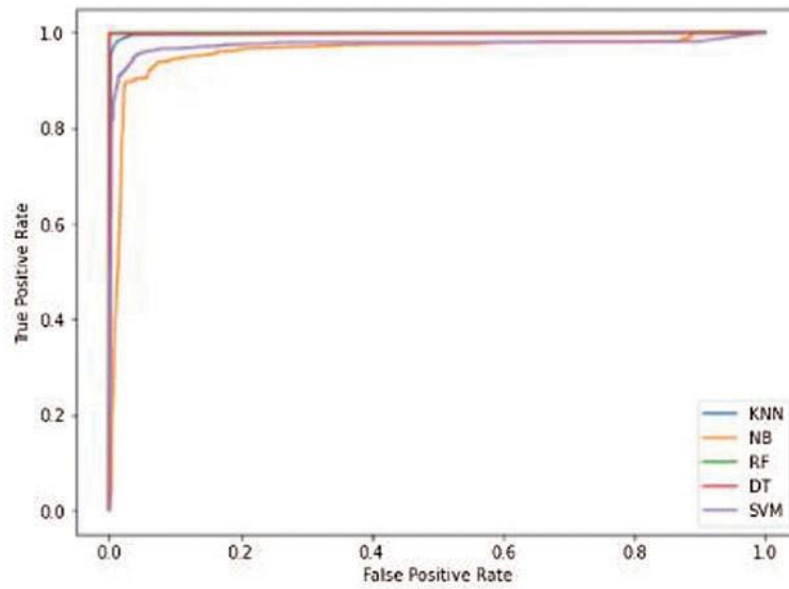


Fig. 3.6: Accuracy curve for BFE

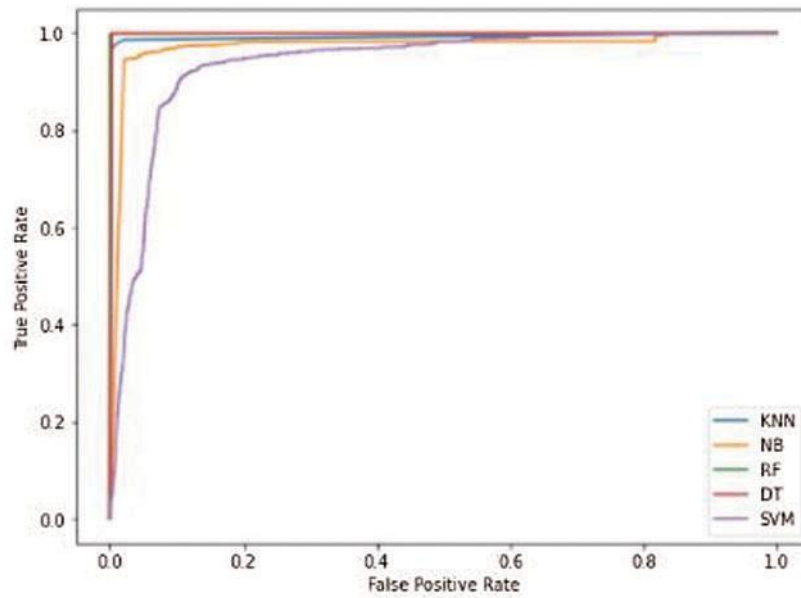


Fig 3.7 : Accuracy curve for RFE

On the other hand, the Chi-Square method relatively achieved low initial accuracy of 48.77% by the NB classifier. It is also observed that the accuracy of the second highest, which is 99.94%, is achieved by the RF classifiers on the Backward feature elimination feature subset. However, it selects the 34 number features that increase the computational time and complexity of the classifier. Therefore, based on the analysis of results, it is reasonable to conclude that the RFE method selects the top-fittest features for the classifiers. Furthermore, the other evaluation metrics that include precision, recall and specificity of the classifiers were also computed using a confusion matrix. The Figs 3.8(a) – 3.8(c) show the comparison of the different classifiers in term of precision, recall and specificity respectively.

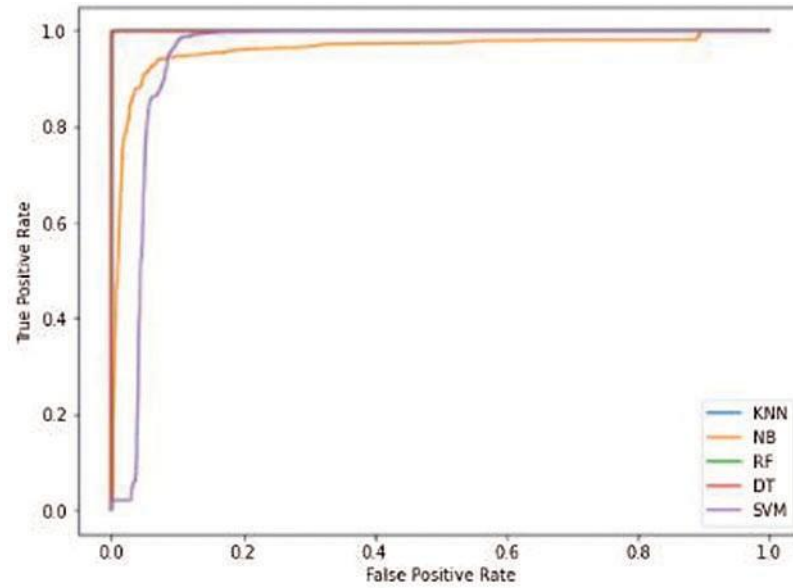


Fig. 3.8: Accuracy curve for Lasso

Table 2: Comparison of experimental results

ML classifiers	Information Gain	Correlation Co-efficient	Chie-Square	Forward Selection	Backward Selection	Recursive elimination	Lasso
SVM	60.19%	80.61%	59.45%	58.60%	59.62%	89.18%	59.79%
KNN	99.57%	99.19%	98.66%	98.50%	98.57%	99.54%	99.41%
NB	85.08%	63.79%	48.77%	84.79%	83.42%	92.12%	69.43%
RF	99.92%	99.91%	99.91%	99.91%	99.94%	99.97%	99.90%
DT	99.88%	99.83%	99.85%	99.80%	99.86%	99.80%	99.83%

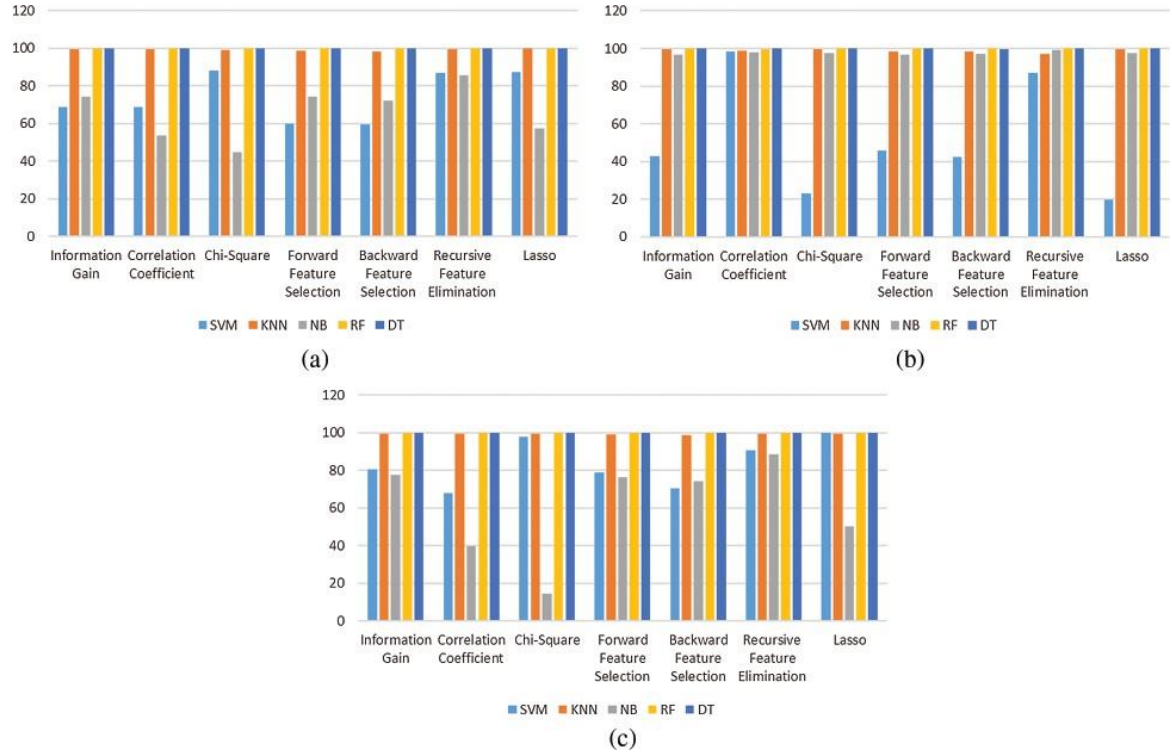


Fig 3.9: (a) Classifiers result in term of precision (b) Classifiers result in term of recall (c) Classifier result in term of specificity

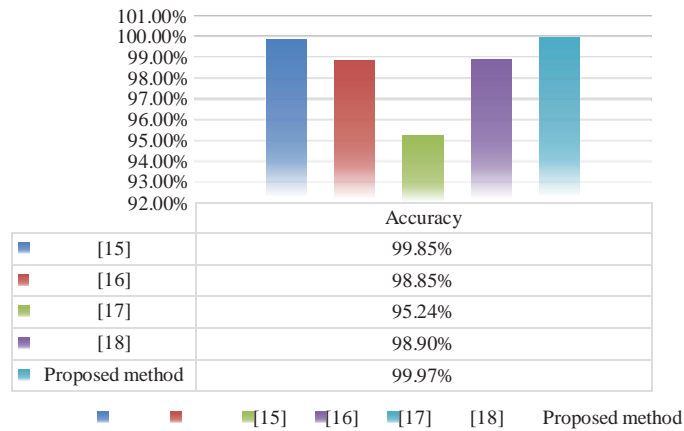


Fig 3.10: Performance comparison of the proposed method with existing solutions in term of accuracy

A comparative analysis of the proposed method with other recent machine learning development for DDoS attack detection in SDN is shown in Fig. 3.9. The accuracy evaluation matrix is used for the comparison purpose. It is clearly observed in Fig 3.9 that the proposed method gives better accuracy as compared to the existing research for the DDoS attack detection in SDN.

We discussed about various approaches of machine learning in SDN by which we can detect different kinds of DDoS attacks. Our project primarily focuses on detection ICMP flood attack, SYN flood attack, UDP flood attack and ICMP smurf attack. We are using two main tools and each of them have their own functionality. One is the RYU controller using which we capture and control traffic, manage flow control for improved network management and application performance. Second is the Mininet , which is a network simulation tool used to create different kinds of virtual networks. A detailed description of both of these tools and how we have used these tools to achieve our end goal is given below,

4.1. RYU Controller

It is one of the SDN controller specially designed for the agility of the network and for managing the higher traffic rate. Ryu includes well-defined software components along with API. Ryu makes the developers develop a new application and manage various other networking devices. Ryu controller is written in Python.

An SDN controller is an application in a software-defined networking (SDN) architecture that manages flow control for improved network management and application performance. The SDN controller platform typically runs on a server and uses protocols to tell switches where to send packets.

SDN controllers direct traffic according to forwarding policies that a network operator puts in place, thereby minimizing manual configurations for individual network devices. By taking the control plane off of the network hardware and running it instead as software, the centralized controller facilitates automated network management and makes it easier to integrate and administer business applications. In effect, the SDN controller serves as a sort of operating system (OS) for the network.

The controller is the core of a software-defined network. It resides between network devices at one end of the network and applications at the other end. Any communication between applications and network devices must go through the controller.

The controller talks with individual network devices using a southbound interface, traditionally one like the OpenFlow protocol. These southbound protocols allow the controller to configure network devices and choose the optimal network path for application traffic.

4.2. Mininet

Mininet is a tool for software-defined networks. It is an emulator of a network and it is used to visualize the switches and application of software-defined networks in a virtualized environment. It is also used to test the software-defined network devices and those using OpenFlow protocols. The switches used in Mininet are OpenFlow switches.

Mininet is majorly used as a learning tool to test, experiment, and learn about software-defined networks. Mininet is preferable because it is very fast and helps us to create customizable topologies. It is also very easy to use.

A Mininet network consists of

ISOLATED HOSTS

A group of user-level processes moved into a network namespace that provide exclusive ownership of interfaces, ports and routing tables.

EMULATED LINKS

Linux Traffic Control (TC) enforces the data rate of each link to shape traffic to a configured rate. Each emulated host has its own virtual Ethernet interface.

EMULATED SWITCHES

The default Linux Bridge or the Open vSwitch running in kernel mode is used to switch packets across interfaces. Switches and routers can run in the kernel or in the user space.

4.3. Detecting DDoS attacks in RYU and Mininet using Machine Learning

4.3.1. Creating Data set

The first step is to start RYU and mininet. Then, to create data set we need two python programs one that can generate traffic in mininet and second running on the RYU controller which will capture the traffic(both legitimate and DDoS) and store it in a CSV file.

We need to collect both legitimate and ddos traffic using RYU for which two files needs to be executed simultaneously on RYU and Mininet. We will generate traffic using our generator programs which we have written using python that will generate normal and ddos traffic. We need to run for both normal and ddos traffic for a considerable amount of time to acquire a good volume of data upon which we can run our machine learning algorithms.

```
Generating traffic ...
.....
Iteration n 1 ...
.....
generating ICMP traffic between h11 and h5 and TCP/UDP traffic between h11 and h1
h11 Downloading index.html from h1
h11 Downloading test.zip from h1
generating ICMP traffic between h6 and h4 and TCP/UDP traffic between h6 and h1
h6 Downloading index.html from h1
h6 Downloading test.zip from h1
generating ICMP traffic between h7 and h1 and TCP/UDP traffic between h7 and h1
h7 Downloading index.html from h1
h7 Downloading test.zip from h1
generating ICMP traffic between h15 and h15 and TCP/UDP traffic between h15 and h1
h15 Downloading index.html from h1
h15 Downloading test.zip from h1
generating ICMP traffic between h15 and h1 and TCP/UDP traffic between h15 and h1
h15 Downloading index.html from h1
h15 Downloading test.zip from h1
generating ICMP traffic between h1 and h6 and TCP/UDP traffic between h1 and h1
h1 Downloading index.html from h1
h1 Downloading test.zip from h1
generating ICMP traffic between h14 and h12 and TCP/UDP traffic between h14 and h1
h14 Downloading index.html from h1
h14 Downloading test.zip from h1
```

Fig 4.1 : Generating Normal Traffic

As we can see here that ICMP and TCP/UDP traffic is generated between different hosts. We can run this for as many iterations as we want to collect data. It is advisable to run it for

atleast 40 iterations to get a good volume of data.

On the other end (controller side) we need to run the collect_benign_traffic.py file simultaneously using RYU controller to aquire the generated traffic and store it in FlowStatsFile.csv.

```
-rw-r--r-- 1 ryu ryu 7.9K Aug 12 2020 RF_controller.py
-rwxrwxr-x 1 ryu ryu 2.4K Jun 1 08:05 RF.py
-rwxrwxr-x 1 ryu ryu 2.1K Jun 1 08:05 SVM.py
-rw-r--r-- 1 ryu ryu 5.5K Aug 12 2020 switch.py
(base) ryu@controller:~/IDS_sdn_network_ddos_detection_using_machine_learning/controller$ ryu-manager collect_benign_traffic.py
loading app collect_benign_traffic.py
loading app ryu.controller.ofp_handler
instantiating app collect_benign_traffic.py of CollectTrainingStatsApp
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Fig 4.2: collecting Normal traffic

Simmilarly, We also need to run collect_ddos_traffic.py and generate_ddos traffic.py on the RYU controller and mininet respectively to collect ddos related dataset and append it with the FlowStatsFile.csv file.

```
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
-----
Performing ICMP (Ping) Flood
-----
Performing UDP Flood
-----
Performing TCP-SYN Flood
-----
Performing LAND Attack
-----
*** Stopping 1 controllers
c0
*** Stopping 23 links
```

Fig 4.3: Generating DDoS traffic'

```
(base) ryu@controller:~/IDS_sdn_network_ddos_detection_using_machine_learning/controller$ ryu-manager collect_ddos_traffic.py
loading app collect_ddos_traffic.py
loading app ryu.controller.ofp_handler
instantiating app collect_ddos_traffic.py of CollectTrainingStatsApp
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Fig 4.4: Collecting DDoS traffic

4.3.2. Running ML Algorithm on Dataset

The next step is to run an ML algorithm on our data set to see what algorithm shows the best results by generating a confusion matrix.

Confusion Matrix is a matrix that allows you to visualize the performance of the classification machine learning models. With this visualization, you can get a better idea of how your machine learning model is performing.

```

.....
confusion matrix
[[ 0 226596]
 [ 0 440285]]
succes accuracy = 66.02 %
fail accuracy = 33.98 %
.....
LEARNING and PREDICTING Time: 0:00:08.646612
.....
K-NEAREST NEIGHBORS ...
.....
confusion matrix
[[226596  0]
 [ 4 440281]]
succes accuracy = 100.00 %
fail accuracy = 0.00 %
.....
LEARNING and PREDICTING Time: 0:01:23.328198
.....
NAIVE-BAYES ...
.....
confusion matrix
[[226596  0]
 [190669 249616]]
succes accuracy = 71.41 %
fail accuracy = 28.59 %
.....
LEARNING and PREDICTING Time: 0:00:01.680314
.....
DECISION TREE ...
.....
confusion matrix
[[226596  0]
 [ 3 440282]]
succes accuracy = 100.00 %
fail accuracy = 0.00 %
.....

```

Fig 4.5: Running ML algorithm on Dataset

4.3.3. Detecting legitimate and DDoS traffic

After finding the most accurate algorithm the next step is to build a controller code using python which will implement the said algorithm to detect legitimate and DDoS traffic. In the scope of the current project we are trying to detect four distinct kinds of DDoS traffics ,those are ICMP Flood, SYN Flood, UDP Flood, ICMP Smurf Attack because those are the most common types of DDoS attack that can negatively affect the flow of information in any networking environment.

First, We run our controller.py file on RYU which will detect whether traffic is legitimate or DDoS.

```

Flow Training ...
.....
confusion matrix
[[ 264  0]
 [ 0 111139]]
succes accuracy = 100.00 %
fail accuracy = 0.00 %
.....
Training time: 0:00:04.428384
instantiating app ryu.controller.ofp_handler of OFPHandler

```

Fig 4.6 : Executing controller on RYU

Second, We run the topology.py file on mininet which will create a virtual linear network topology with 6 switches and 18 hosts , each switch conneted to the next switch and 3 hosts.

4.3.3.2. Detecting DDoS Traffic

Before, seeing how the controller detects different kinds of DDoS attacks we need to learn about how are we going to simulate said attacks on the mininet. The said task is done by using a software tool known as hping3.

The tool hping3 allows you to send manipulated packets. This tool allows you to control the size, quantity and fragmentation of packets in order to overload the target and bypass or attack firewalls. Hping3 can be useful for security or capability testing purposes, using it you can test firewalls effectively and if a server can handle a big amount of packets. Below you will find instructions on how to use hping3 for security testing purposes.

1) Flood using SYN packets

The aim of SYN flood is sending lots of SYN packets to the server and ignoring SYN+ACK packets returned by the server. This causes the server to use their resources for a configured amount of time for the possibility of the expected ACK packets arriving. If an attacker sends enough SYN packets, this will overwhelm the server because servers are limited in the number of concurrent TCP connections. If the server reaches its limit, it cannot establish new TCP connections until the existing connections which are in the SYN-RCVD state timeout.

Command : `sudo hping3 lacampora.org -q -n -d 120 -S -p 80 --flood --rand-source`

Where:

Lacampora.org: is the target

-q: brief output

-n: show target IP instead of host.

-d 120: set packet size

--rand-source: hide IP address.



```
root@mininet-vm:~/IDS_sdn_network_ddos_detection_using_machine_learning/mininet
# hping3 -S -V -d 120 -w 64 -p 80 --rand-source --flood 10.0.0.1
using h13-eth0, addr: 10.0.0.13, MTU: 1500
HPING 10.0.0.1 (h13-eth0 10.0.0.1): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

Fig 4.10: Performing syn attack

```

-----
legitimate traffic ...
-----
ddos traffic ...
victim is host: h1
-----

```

Fig 4.11 : DDoS traffic on host 1 shown on controller window

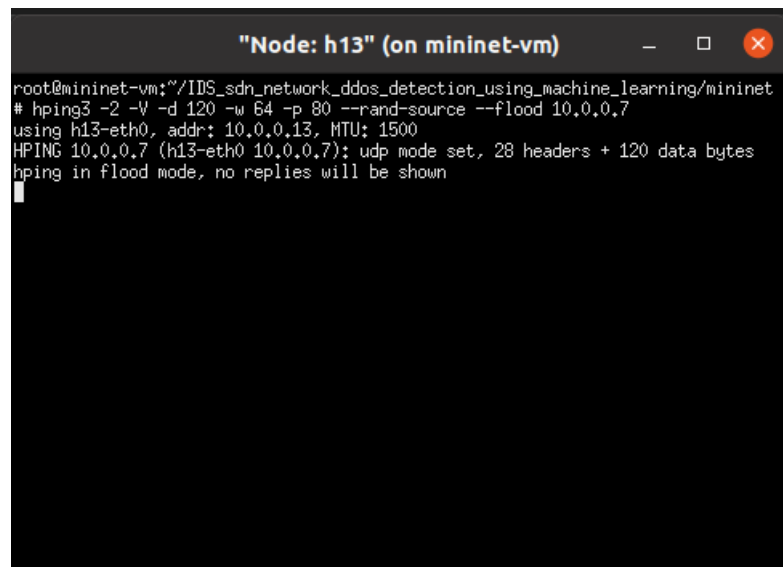
2) UDP Flood:

UDP is a protocol which does not need to create a session between two devices. In other words, no handshake process required.

A UDP flood does not exploit any vulnerability. The aim of UDP floods is simply creating and sending large amount of UDP datagrams from spoofed IP's to the target server. When a server receives this type of traffic, it is unable to process every request and it consumes its bandwidth with sending ICMP "destination unreachable" packets.

Command: `hping3 -flood --rand-source --udp -p TARGET_PORT TARGET_IP`

- flood: sent packets as fast as possible
- rand-source: random source address
- udp: UDP mode
- p –destport: destination port (default 0)



```

"Node: h13" (on mininet-vm)
root@mininet-vm:~/IDS_sdn_network_ddos_detection_using_machine_learning/mininet
# hping3 -2 -V -d 120 -w 64 -p 80 --rand-source --flood 10.0.0.7
using h13-eth0, addr: 10.0.0.13, MTU: 1500
HPING 10.0.0.7 (h13-eth0 10.0.0.7): udp mode set, 28 headers + 120 data bytes
hping in flood mode, no replies will be shown

```

Fig 4.12: Performing UDP Flood

```
legitimate traffic ...
```

```
ddos traffic ...  
victim is host: h7
```

```
ddos traffic ...  
victim is host: h7
```

Fig 4.13: Detecting UDP Flood

3) ICMP Flood:

ICMP(Internet Control Message Protocol) are connectionless protocols like UDP. ICMP is used for sending error messages and operational information from network devices.

Like UDP flood, ICMP floods does not exploit any vulnerability. Just sending any type of ICMP packets continuously makes server overwhelmed from trying to process every request.

Command: `hping3 -flood -rand-source -i 1 -p TARGET_PORT TARGET_IP`

```
"Node: h13" (on mininet-vm)
root@mininet-vm:~/IDS_sdn_network_ddos_detection_using_machine_learning/mininet
# hping3 -i 1 -V -d 120 -w 64 -p 80 --rand-source --flood 10.0.0.12
using h13-eth0, addr: 10.0.0.13, MTU: 1500
HPING 10.0.0.12 (h13-eth0 10.0.0.12): icmp mode set, 28 headers + 120 data bytes
hping in flood mode, no replies will be shown
```

Fig 4.14 : Performing ICMP flood

```
legitimate traffic ...
```

```
legitimate traffic ...
```

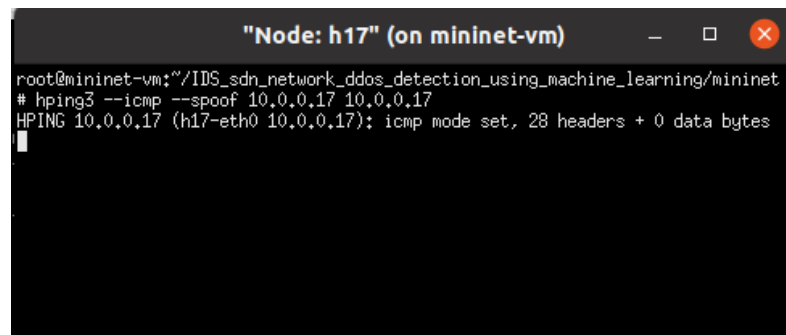
```
ddos traffic ...  
victim is host: h12
```

Fig 4.15: Detecting ICMP Flood

4) Smurf Attack:

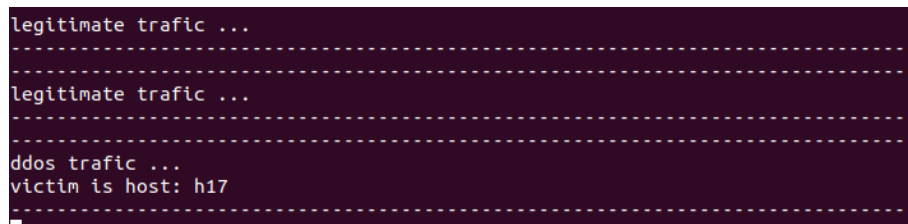
Attacker chooses some intermediary sites as an amplifier, then sends the huge amount of ICMP(ping) requests to the broadcast IP of these intermediary sites. By the way, these packets have the source IP addresses point towards the target. Intermediary sites deliver the broadcast to all the hosts on their subnet. Finally, all hosts reply to target IP.

Command: `hping3 --icmp --spooof TARGET_IP BROADCAST_IP`



```
"Node: h17" (on mininet-vm)
root@mininet-vm:~/IDS_sdn_network_ddos_detection_using_machine_learning/mininet
# hping3 --icmp --spooof 10.0.0.17 10.0.0.17
HPING 10.0.0.17 (h17-eth0 10.0.0.17): icmp mode set, 28 headers + 0 data bytes
```

Fig 4.16: Performing Smurf attack



```
legitimate traffic ...
-----
legitimate traffic ...
-----
ddos traffic ...
victim is host: h17
-----
```

Fig 4.17: Detecting Smurf attack

5.1. Conclusion

Although the SDN makes considerable advancements in networking, it faces several security issues, where the most common security issue for the SDN is DDoS attacks. The SDN controller is the centralized control of the whole network and it becomes more vulnerable to DDoS attacks can reach there. Hence, the intelligent detection of DDoS attacks in the SDN is needed. In response to this problem, this paper presents a comparative analysis of different machine learning classifiers based on the optimal subset of features for the early and accurate detection of DDoS attacks over the SDN. The combination of the machine learning classifiers and the advantages of the SDN protects the SDN controller from DDoS attacks.

Furthermore, the extraction and selection of optimal features for the machine learning-based models are also crucial for accurately detecting an attack. The experimental results prove that the RF classifier on recursive feature elimination ranked features subset achieves good results for detecting attacks in the SDN controller. However, the resources consumption of the SDN controller increases and the detection accuracy of the DDoS attack decreases when the network is under a larger-scale network traffic. Furthermore, the use of an irrelevant and many features also increase the SDN control's workload, which may affect the controller's efficiency.

5.2. Future Work

In the future, these machine learning classifiers and feature selection techniques would also be used to detect the other attack classes such as smurf, Probe, R2L and U2R, in the SDN. Although in the environment of a single controller, this work is well for detecting DDoS, it may fail to detect the attack traffic in a multi-controller environment. So, in the future, these models are also evaluated to detect attacks in a multi-controller context.

Bibliography

- [1] Y. Zhang, L. Cui, W. Wang and Y. Zhang, “A survey on software defined networking with multiple controllers,” *Journal of Network and Computer Applications*, vol. 103, no. 3, pp. 101–118, 2018.
- [2] P. Visu, L. Lakshmanan, V. Murugananthan and M. V. Cruz, “Software-defined forensic framework for malware disaster management in internet of thing devices for extreme surveillance,” *Computer Communications*, vol. 147, no. 5, pp. 14–20, 2019.
- [3] E. Molina and E. Jacob, “Software-defined networking in cyber-physical systems: A survey,” *Computers & Electrical Engineering*, vol. 66, no. 11, pp. 407–419, 2018.
- [4] A. Mondal, S. Misra and I. Maity, “AMOPe: Performance analysis of openflow systems in software- defined networks,” *IEEE Systems Journal*, vol. 14, no. 1, pp. 124–131, 2019.
- [5] M. Conti, C. Lal, R. Mohammadi and U. Rawat, “Lightweight solutions to counter DDoS attacks in software defined networking,” *Wireless Networks*, vol. 25, no. 5, pp. 2751–2768, 2019.
- [6] C. B. Zerbini, L. F. Carvalho, T. Abrao and M. L. Proenca Jr, “Wavelet against random forest for anomaly mitigation in software-defined networking,” *Applied Soft Computing*, vol. 80, pp. 138–153, 2019.
- [7] Y. Xu and Y. Liu, “DDoS attack detection under SDN context,” in *35th Annual IEEE Int. Conf. on Computer Communications*, San Francisco, CA, USA, pp. 1–9, 2016.
- [8] R. T. Kokila, S. T. Selvi and K. Govindarajan, “DDoS detection and analysis in SDN-based environment using support vector machine classifier,” in *Sixth Int. Conf. on Advanced Computing*, Chennai, India, pp. 205–210, 2014.
- [9] H. Wang, L. Xu and G. Gu, “Floodguard: A dos attack prevention extension in software-defined networks,” in *45th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, Rio de Janeiro, Brazil, pp. 239–250, 2015.
- [10] K. S. Sahoo, A. Iqbal, P. Maiti and B. Sahoo, *A Machine Learning Approach for Predicting DDoS Traffic in Software Defined Networks*. Bhubaneswar, India: International Conference on Information Technology, pp. 199–203, 2018.
- [11] N. Meti, D. G. Narayan and V. P. Baligar, “Detection of distributed denial of service attacks using machine learning algorithms in software defined networks,” in *Int. Conf. on Advances in Computing, Communications and Informatics*, Udupi, India, pp. 1366–1371, 2017.
- [12] S. Shin, V. Yegneswaran, P. Porras and G. Gu, “Avant-guard: Scalable and vigilant switch flow management in software-defined networks,” in *Proc. of the 2013 ACM SIGSAC Conf. on Computer & Communications Security*, New York, United States, pp. 413–424, 2013.
- [13] Z. Chen, F. Jiang, Y. Cheng, X. Gu, W. Liu *et al.*, “XGBoost classifier for DDoS attack detection and analysis in SDN-based cloud,” in *IEEE Int. Conf. on Big Data and Smart Computing*, Shanghai, China, pp. 251–256, 2018.
- [14] M. Latah and L. Toker, “Towards an efficient anomaly-based intrusion detection for software-defined networks,” *IET Networks*, vol. 7

- [15] A. B. Dehkordi, M. Soltanaghaei and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SDN," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 2383–2415, 2021.
- [16] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang *et al.*, "A new framework for DDoS attack detection and defense in SDN environment," *IEEE Access*, vol. 8, pp. 161908–161919, 2020.
- [17] J. Ye, X. Cheng, J. Zhu, L. Feng and L. Song, "A DDoS attack detection method based on SVM in software defined network," *Security and Communication Networks*, vol. 2018, no. 4, pp. 11–23, 2018.
- [18] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy *et al.*, "An evolutionary SVM model for DDOS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 132502– 132513, 2020.
- [19] J. A. P. Díaz, I. A. Valdovinos, K. K. R. Choo and D. Zhu, "A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning," *IEEE Access*, vol. 8, pp. 155859–155872, 2020.
- [20] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras and V. Maglaris, "Combining open- flow and sflow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, no. 2, pp. 122–136, 2014.
- [21] L. Schehlmann and H. Baier, "COFFEE: A concept based on openflow to filter and erase events of botnet activity at high-speed nodes, information. 2013-informatik angepasst an mensch," *Organization und Umwelt*, vol. 220, pp. 2225–2239, 2013.
- [22] J. Ashraf and S. Latif, "Handling intrusion and DDoS attacks in software defined networks using machine learning techniques," in *National Software Engineering Conf.*, Rawalpindi, Pakistan, pp. 55–60, 2014.

