IDENTIFICATION OF ATTACKS ON SOFTWARE-DEFINED NETWORK USING MACHINE LEARNING APPROACH

Thesis submitted in fulfillment of the requirements for the Degree of

DOCTOR OF PHILOSOPHY

BY

NISHA AHUJA

(E17SOE831)

School of Computer Science Engineering and Technology

BENNETT UNIVERSITY

(Established under UP Act No 24, 2016)

Plot Nos 8-11, Tech Zone II,

Greater Noida-201310, Uttar Pradesh, INDIA

September, 2022

IDENTIFICATION OF ATTACKS ON SOFTWARE-DEFINED NETWORK USING MACHINE LEARNING APPROACH

Thesis submitted in fulfillment of the requirements for the Degree of

DOCTOR OF PHILOSOPHY

BY

NISHA AHUJA

(E17SOE831)

School of Computer Science Engineering and Technology

BENNETT UNIVERSITY

(Established under UP Act No 24, 2016)

Plot Nos 8-11, Tech Zone II,

Greater Noida-201310, Uttar Pradesh, INDIA

September, 2022

DEDICATED TO MY LOVING FAMILY

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the Ph.D. thesis entitled **Identification of attacks on Software-Defined Network Using Machine Learning Approach** submitted at **Bennett University, Greater Noida, India**, is an authentic record of my work carried out under the supervision of **Dr. Tapas Badal, Dr. Gaurav Singal, and Prof. Debajyoti Mukhopadhyay**. I have not submitted this work elsewhere for any other degree or diploma.

(Signature of the Scholar)

Nisha Ahuja
Enrollment No. E17SOE831
School of Computer Science Engineering and Technology
Bennett University, Greater Noida, India
Date

SUPERVISOR'S CERTIFICATE

This is to certify that the work reported in the Ph.D. thesis entitled **Identification of attacks on Software-Defined Network Using Machine Learning Approach**, submitted by **Nisha Ahuja** at **Bennett University, Greater Noida, India**, is a true record of her original work completed under my supervision. This work has not been submitted anywhere for any other degree.

(Signature of Supervisor)
Dr. Tapas Badal
Asst. Professor, Department of CSE
School of CSET, Bennett University

Date:

(Signature of Supervisor)
Dr. Gaurav Singal
Asst. Professor, Department of CSE
Netaji Subhash University of Technology, Delhi
Date:

(Signature of Supervisor)
Prof. Debajyoti Mukhopadhyay
Date:

ACKNOWLEDGEMENT

Ph.D. is a very hard journey that cannot be completed without the support of our well-wishers. Firstly, I would Thank my supervisor Dr. Tapas Badal, who has shown confidence in my work and given constructive feedback on the thesis. Without his support, this work would not have been possible. This thesis work would not have been in the present form without the constant support from my Co-Supervisor Dr. Gaurav Singal whose productive criticisms significantly improved the thesis. It is impossible to cultivate ideas more clearly without his critical support. His feedback also assisted me in broadening my ideas in a variety of aspects.

I am also thankful to the other supervisor Prof. Debajyoti Mukhopadhyay, who has been supportive of my career goals and who worked actively to provide me with his support and time to pursue those goals. I am especially indebted to Dr. Deepak Garg, Head of the Department of Computer Science, who taught me the importance of time management to meet the deadlines. Each of the members of my Dissertation Committee has provided me with extensive personal and professional guidance and taught me a great deal about both scientific research and life in general. The non-teaching employees were always helpful and assist me to the best of their abilities.

Most importantly, I wish to thank my loving parents who provide unending inspiration that helped me survive this journey. I would like to express my gratitude to my Husband whose cooperation in this journey is respected and admired. My son Hridyansh, always revitalises me with his lively and enthusiastic talks. Thank you all again for your essential warm-hearted support.

(Nisha Ahuja)

TABLE OF CONTENTS

ABSTRACT

SDN is a communication technology defined by a software program that manages network traffic routing and configuration. In contrast, the current network architecture controls traffic by configuring the various network elements remotely. The SDN architecture is centralized in nature such that data plane and control plane within a networking device are segregated. The control plane can be thought of as the mind of the network whereas the data plane is adhering to the controller's decisions. Examples of SDN controllers include FloodLight, Ryu, Pox, OpenDayLight, Nox, etc. which are open source and incorporate a set of APIs for building network applications.

Many researchers have worked on the detection of attacks and the categorizing of network traffic into benign and malicious categories. Existing DDOS attack detection research is based on threshold-based detection on the count of incomplete connections made, the number of queries made per user, traffic rate, and the total time of flow duration. Other techniques include computing the feature tensors for the construction of benign and malicious vectors and comparing these vectors to a threshold parameter for attack detection. Other techniques include the use of a Markov model on a network graph, a tensor-based technique for calculating the entropy of TCP layer attributes, randomness in different traffic features (such as Destination IP address, Source IP address, Protocol type, TCP flags, Destination Port, Source Port, and Packet size) and Machine learning (ML) based approaches. The techniques discussed above detect attacks by comparing a specific value as in threshold-based approach, which is impractical in a large network. Some of them trained the deep learning model on a traditional dataset which are not created in SDN environment. The detection method employed is computationally time-consuming, and the experimental setup is not adequately described.

As a result of the identified research gaps, the thesis work proposed various machine learning and deep learning algorithms, which are discussed in the following chapters. The hybrid models of Support vector classifier with Random Forest, CNN-LSTM, and Stacked Auto-Encoder (SAE) with Multi-layer Perceptron (MLP) achieved the highest accuracy for DDoS and ARP-based attack detection. The authors also generated DDoS attack traffic and ARP poisoning

attack dataset. The annotated traffic dataset is utilized by the machine learning classifiers for network traffic classification into benign and malicious classes. The network traffic is generated by collecting the traffic statistic from the various nodes present in the topology. The topologies are designed by emulating the industrial network scenario using mininet. Various statistics are collected from the network switches within a monitoring interval of 30 seconds.

In this thesis, we inspected the role of ML classifiers to classify the DDOS attack and benign traffic. From the total of sixteen features, nine are extracted from the switches (namely packet count, byte count, duration nsecs, duration secs, source IP, destination IP, txbytes, rxbytes, and port number) while others are calculated based on the extracted features. The created dataset is a collection of features that encompass more than 100k traffic instances in the DDoS attack traffic datasets.

We also investigated the role of machine learning classifiers in classifying ARP-based attack and benign traffic. The SDN dataset for ARP-based attack has been created. There are total of seventeen features in the dataset (such as switch-id, Round trip time, Time to live (TTL), Packet loss, Number of Packet_in messages, source MAC address in ethernet, destination MAC address in ARP, source IP, destination IP, ping statistics, and operation code, etc.) The created dataset includes 1,34,000 rows and ML algorithms are trained and tested on the dataset. The dataset is used by different machine learning algorithms for traffic classification among ARP Poisoning, Flood attack, and benign traffic.

We also investigated DDOS attack detection using the Deep learning technique. In comparison to the work of attack detection using machine learning approaches, the deep learning technique provided significant performance. The hybrid model of Self-auto-Encoder and Multilayer Perceptron (SAE-MLP) achieved the highest classification accuracy of all Deep Learning models tested. The four different publicly available datasets CICIDS2017, CIC-DoS, and CSE-CIC-IDS-2018 are also used to evaluate the generated dataset for DDoS attack traffic classification. After evaluation, it is found that the proposed Dataset classifies the traffic with the highest accuracy.

The results obtained in DDOS attack traffic classification using machine learning techniques were quite promising and provide an accuracy of 98.8%. These results are further improved by

applying deep learning techniques for DDOS attack traffic classification and attain an accuracy of 99.73%. The hybrid model of CNN-LSTM provides significant classification accuracy of 99.73% among the different ML algorithms used for traffic classification.

LIST OF ACRONYMS

TCP	Transport Control Protocol
MAC	Media Access Control
IP	Internet Protocol
SDN	Software Defined Networking
DDOS	Distributed Denial of Service
ARP	Address Resolution Protocol
CIC-IDS	Canadian Institute for Cybersecurity-Intrusion Detection System
CIC-DoS	Canadian Institute for Cybersecurity-Denial of Service
CSE-CIC-IDS	Communications Security Establishment-Canadian Institute for Cybersecurity-Intrusion Detection System
CNN-LSTM	Convolution Neural Network-Long Short Term Memory
SAE-MLP	Stacked Auto Encoder-Multi-Layer-Perceptron
SOM	Self-Organizing Map
API	Application Programming Interface
WAN	Wide Area Network
Json	JavaScript Object Notation
ONF	Open Networking Foundation
SVC-RF	Support Vecot Classifier-Random Forest
NFV	Network Function Virtualization
RNN	Recurrent Neural Network
DNS	Domain Name System
VOIP	Voice Over IP
AUC	Area under the ROC Curve
NSL-KDD	Network Security Laboratory-Knowledge Discovery Data Mining
LDA	Linear Discriminant Analysis
SVM	Support Vector Machine
5G	Fifth Generation
NAD	Network Anomaly Detector
PCA	Principle Component Analysis
BHP	Burst Header Packet

KB	KiloBytes
SHDA	Secure hash Decryptable Analysis
TTL	Time to Live
ACL	Access Control List
DARPA	Defense Analysis Researach Project
ISCX	Intelligent Science Center for Xtraction
IETF	Internet Engineering Task Force
FORCES	Forwarding and Control Element Separation

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

CHAPTER 1
INTRODUCTION

Software-Defined Networking (SDN) is a next-generation network in which the networking elements are programmed. The SDN architecture allows rigid and inflexible network devices to be programmed. Traditional networking devices are fixed, inflexible, and designed in such a way that we must write code to test any functionality. If there are multiple devices, each one must be manually configured. SDN no longer contains any of these requirements. It is a novel networking paradigm that has redefined the networking domain and proven to be a significant milestone for network engineers and administrators. In the following section, the concept of SDN is discussed along with its role in detecting the different attacks.

1.1 Software-Defined Networking

Software-Defined Networking (SDN) has emerged as the revolution in the networking industry. The SDN design allows traditional network devices to be programmable. Many of the industries are using SDN in their data centers. There were many R&D networks based on Open-Flow switches from NEC and Hewlett-Packard but that exists in academic settings only. SDN originates from the time when it was first time used for telephone line where the architecture of separating the two planes was first implemented. The Internet Engineering Task Force (IETF) has taken a step forward in this direction. IETF then worked on developing the network standard in 2004 known as Forwarding and Control Element Separation (FORCES) soon after which the platform was used in data centers. There are many more standards from IETF that work for the same task of separating control and data plane namely Linux Net Link based on IP as Protocol. The concepts did not prove to be workable because of two reasons, First & foremost, the separation of two planes was found to be insecure and attack prone due to the centralized controller. Secondly, vendors were not in favor of such change as this will create a new level of services, and thus new APIs need to be introduced. The Ethane project at Stanford University's Computer

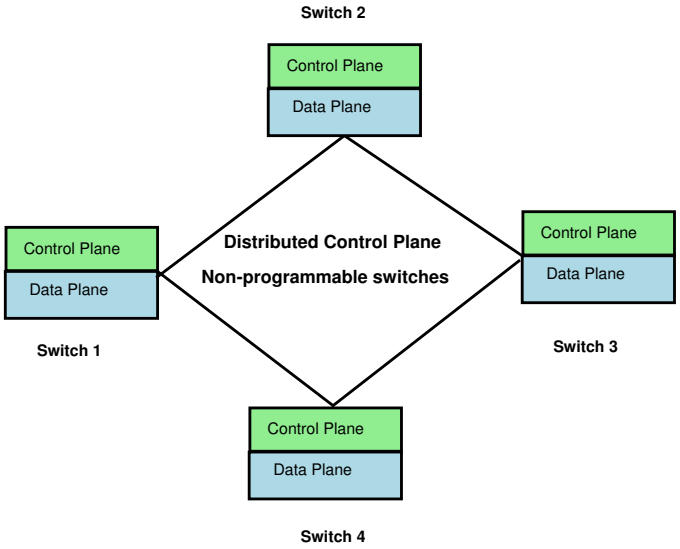

Figure 1.1: Visualization of the Traditional/Present Network in an architectural diagram.

Science Department pioneered the use of open-source software. In 2008, an OpenFlow API was introduced for the first time. NOX, a network operating system, was created in the same year.

In 2007, independent researchers filed many patent applications outlining SDN applications, network operating systems, network equipment data processing units, and a method for virtualization. At Stanford, work on OpenFlow proceeded, with testbeds being built to examine the protocol's utility. Martin Casado, a researcher at Stanford University, invented SDN. In 2011, he invented open flow, which is an SDN protocol. SDN can be used for cloud computing architecture where till now compute, storage and infrastructure are only available on-demand but after the technology of SDN & its complementary technology NFV, Network is also available on demand.

The architecture diagram depicting the traditional network and SDN is shown in the Figure 1.1, 1.2. The devices in the network are hard to configure as the two planes in the device i.e., Data Plane and Control Plane are tightly coupled. These two planes are linked in such a way that they cannot be programmed and are thus rigid. But, Software-Defined Network is based on the concept of separating the data plane and Control Plane. The Control Plane is understood as the one responsible for making all the decisions in the network. We can see the control plane in SDN like the brain in the human body and Data Plane is known for its functionality of just forwarding the network traffic. SDN architecture is described with three planes as follows:

1. Application Plane: It is the plane that is responsible for the development of the applica-

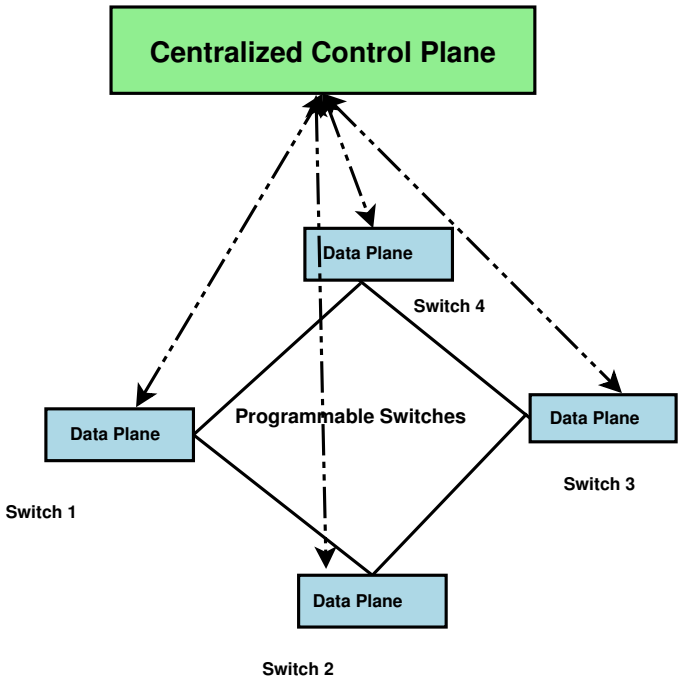

Figure 1.2: Visualization of the Software-Defined Networking in an architectural diagram.

tions which will be executed by the control Plane [4]. Applications written in languages such as C, Java, and Python can be deployed at the control plane. If any user wants to retrieve some information from the control plane, it can be retrieved as a json file or a text file.

2. Control Plane: This is the plane where the controller exists. It is the brain of all the switching devices. The controller uses the south-bound interface to communicate with Data Plane devices. It will execute the code from the application Plane which will affect all the devices in Data Plane.

3. Data Plane: It is the plane in which all the switching devices reside. They are just the traffic forwarders and work as inactive terminals and follow the decision made by the controller.

This architectural change has brought a lot of flexibility to the network. Unlike traditional networks [4] where the control plane is coupled with the data plane in a device. In SDN, the isolation of the control plane from the Data Plane makes the network programmable. Due to the architectural differences in the networking paradigms, the traditional way of forwarding network packets differs from the SDN. The traditional method of traffic forwarding has been explained as under:

1. Traditional switches include a Control Plane and a Data Plane.
2. When the switch is turned on, the MAC Table is empty.
3. MAC ID and Port Number are updated in the MAC Table by the Control Plane. This data is extracted from the packet.
4. Control Plane continues to build/update the MAC Table.
5. When a packet reaches the destination, the data plane inspects the MAC table. Based on the matching destination MAC address, the switch routes it to the appropriate port from the MAC table.

Since SDN is based on network programming, the method of forwarding network traffic is as follows:

1. The Openflow protocol version is configured in the SDN Controller and Switch.
2. The switch establishes contact with the SDN Controller.
3. The predefined Openflow rule (TABLE MISS ENTRY) is added to the switch's Flow table by SDN Controller.
4. The TABLE MISS ENTRY openflow rule fits all packets arriving for the first time and sends them to the Controller.
5. When the packet is received at the destination, it is matched with TABLE MISS ENTRY and forwarded to the Controller (PACKET IN Message).
6. The controller accepts the packet and uses it to construct the Switch tables.
7. The controller keeps adding incoming flows to the switch.
8. Flows are used to construct the Switch data path. As a result, when the packet arrives, it is compared to the Flow table and forwarded to the appropriate port.

Aside from such distinctions between the two networking paradigms, SDN faces many challenges, including the same security threats as traditional networks. The security attacks can occur on any of the planes in the SDN architecture. An emulator or simulator is required to carry out the various experimental tasks in SDN. To perform more accurate experiments, emulators are used in a variety of experiments.

1.2 Motivation

In the realm of research, Software Defined Networking (SDN) opens up a plethora of possibilities. Its novel architecture presents several obstacles as well as future directions for communication enhancement. Security is an important aspect when basic functions are to be delivered effectively. SDN's security domain encompasses a several features introduced by its innovative design and by providing new solutions to old network vulnerabilities. In SDN, the controller provides programming capabilities to automate the solution provisioning process. However, if the attacker erroneously propagates false information about the connected host, even the controller will be unable to make the correct judgment. One such attack is the ARP Poisoning attack in SDN, which has received little attention from prior researchers. Another attack of Distributed Denial of Service (DDoS) has also been a serious threat in SDN, for which different strategies exist in the traditional networks. But little attention from prior researchers has been done to SDN. This motivates the author to propose a unique solution for the prevention of ARP Poisoning, ARP Flood, and DDOS attacks. The different reasons due to which the research work in the thesis has been done are mentioned below:

- The latest trends in cloud computing and other fields have created a demand for the creation of a flexible network, which traditional networks do not provide. SDN, on the other hand, can provide network on-demand services, which encourages further research into the concept.
- SDN's security confronts various issues because it is handled by a centralized controller, prompting researchers to focus on network security.
- SDN's performance should not be harmed as a result of security concerns, because it delivers various advantages.
- SDN security is critical because it will be used by cloud service providers.

1.3 Contribution

In this thesis, we have investigated mainly DDOS, ARP Poisoning, and ARP Flood attack. The attacks have been detected using Machine Learning Algorithms with promising results. The following are the research contributions, as well as the publications in which they are mentioned:

- First and foremost, we conduct a thorough review of the literature on security issues in SDN. This was explored in Chapter 2 of the book. According to our findings, the majority of security issues in SDN are due to weaknesses in protocols such as TCP, ARP. We also see that the majority of the security solutions presented are cryptographic or static binding-based. These solutions are computationally expensive. Our main goal was to provide non-cryptographic, lightweight solutions. The proposed solutions are presented in reputed Conferences and Journals.

- DDOS attack Dataset generated: There are several DDOS attack datasets available, however, they've all been evaluated in traditional network environments. Others created in the SDN environment have not been given access. The authors created a dataset in the SDN environment, which is available in the Mendeley repository for public usage in Mendeley repository [1].

- ARP Poisoning and Flood attack Dataset: There is little attention given by the researchers towards ARP-based attacks attack. There is very less public available datasets for these attacks in the SDN environment. The authors generated ARP Poisoning and Flood attack dataset in the SDN environment, which is also available in the Mendeley repository [2] for public usage.

- DDOS attack traffic classification: The methods of DDOS attack detection has been investigated by previous researchers but most of them used either statistical or ML-based solution. Those solutions seem infeasible in the SDN environment. The proposed research work focuses on providing a simple and efficient way of detecting the attack. The work makes use of the above generated DDOS attack Dataset and trains an ML algorithm for traffic classification into Benign and DDOS attack traffic classes respectively.

- ARP Poison and flood attack Traffic Classification: Previous researchers experimented with ARP Poisoning and Flood attack detection systems. The majority of them relied on statistical or cryptographic solutions. However, in an SDN environment, the solutions appear to be infeasible. The proposed research focuses on developing simple and effective

[1] https://data.mendeley.com/datasets/jxpfjc64kr/1
[2] https://data.mendeley.com/datasets/yxzh9fbvbj/1

methods for detecting the attack. The work uses the above-mentioned ARP poisoning and ARP Flood attack dataset to build a machine learning algorithm for traffic classification into benign and ARP Poisoning and ARP Flood attack traffic classes.

1.4 Thesis Structure

The structure of the thesis is as follows. Chapter2 contains a detailed description of the existing state-of-the-art research in the area of DDOS attack detection and ARP-based attack detection. This chapter also identifies SDN plane's vulnerabilities to various attacks. The proposed work done for the detection and classification of DDOS attacks is discussed in Chapter III. Chapter IV describes the proposed detection and classification mechanism scheme for ARP Poison and ARP Flood attacks in SDN. The chapter also discusses the CPU utilization and time taken for attack detection to efficiently present the results. Chapter V describes the work on DDoS attack detection and traffic classification using Deep Learning techniques. Finally, Chapter VI ends with the conclusion. The author's national and international publications are also mentioned at the end.

CHAPTER 2
LITERATURE SURVEY

This chapter examines the literature about our research objectives. Two security attacks namely DDoS and ARP-related attacks in SDN are the primary focus of this research. This chapter is based on existing potential threats, available research, and identified gaps. This analysis will be useful in the upcoming chapters, which will provide solutions to various threats. In this chapter, state-of-the-art research available on security attacks in SDN has been discussed. The existing research in DDoS and ARP Poisoning and Flood attack detection are discussed below:

2.1 Attacks in SDN

The attacks which occur at different SDN planes are outlined in Figure 2.1 which includes DDoS attack, ARP-Spoofing attack, side-channel attack, and many others. Although SDN provides many benefits in terms of programmability, flexibility, and simplicity. But it all comes at the expense of security if not taken care of properly. As the controller is centralized and derives the whole network it is faced with the issue of security. So, security needs to be taken care of at all the planes of SDN. The different types of attacks that can occur in SDN are shown in Figure 2.1. The security attacks which affect the SDN are as follows:

1. Attacks at Application Plane: There can be various attacks that are possible at the application plane of SDN. As the application plane uses the applications from a third party, the applications developed can be malicious. If they are malicious then it can affect the whole network starting from the controller to the switches and thus entire network can be attacked [5]. Some of the attacks which can occur at the application plane are:

   (a) API Exploitation: In this type of attack the API which is active between the controller and application layer is exposed to attack by the attackers. If the API is exploited by the attackers, then the control layer [6] is vulnerable to attacks.

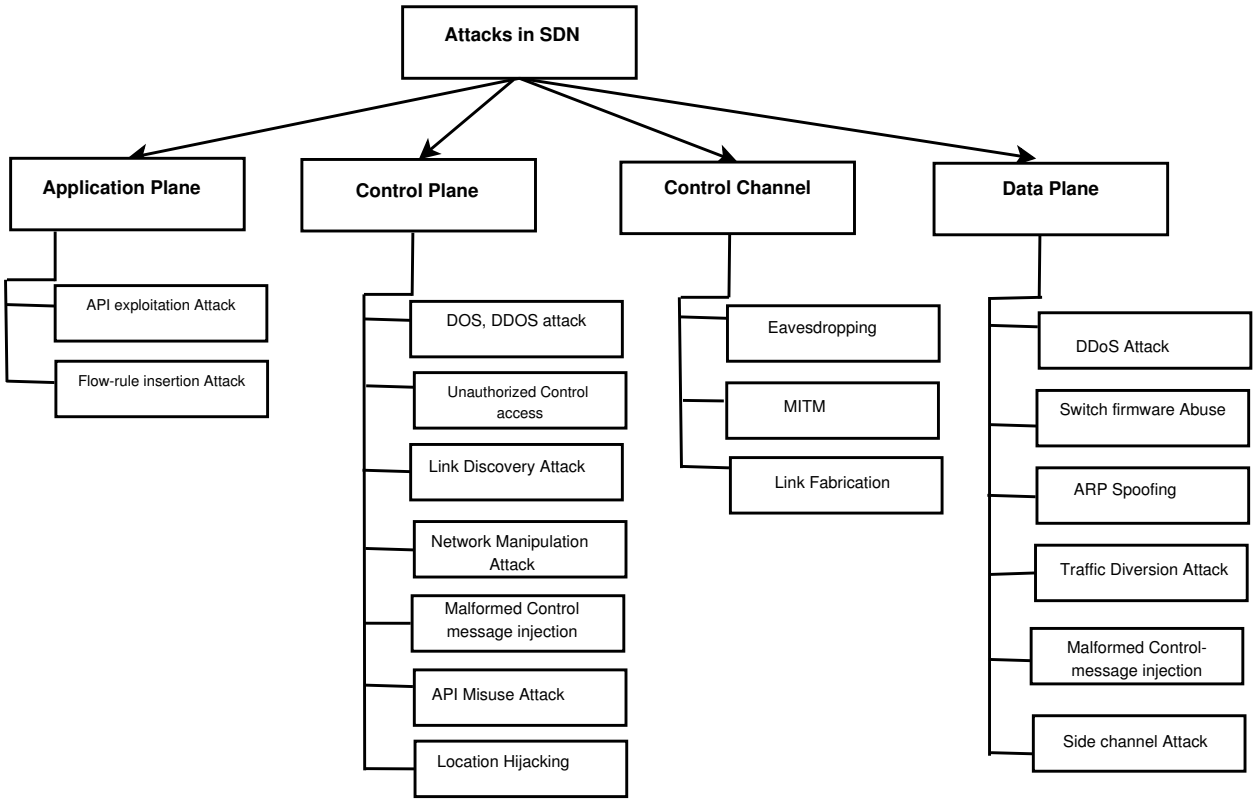

Figure 2.1: SDN Attack taxonomy depicts the attacks which can occur on different Planes.

(b) Flow-rule insertion attack: Third-party applications at the application layer manipulate the flow tables and insert new rules into the flow table, causing network manipulation.

2. Attacks at Control Plane: Control Plane executes the applications built for configuring the data plane. The brain of SDN i.e., the Controller is the one that will decide about traffic routing and other configurations. But if that is compromised then it is very difficult for the network to work properly. The various attacks that can occur at Control Plane are:

   (a) DoS: This type of attack occurs when the control plane resources are filled with malicious requests in such a way that if any legitimate requests come it cannot answer the request.

   (b) Network Manipulation attack: In this attack, the third party which is programming the network can create a code in which false routes can be added to the network. This type of attack results in the entire SDN network failure.

3. Attacks on Control Channel: The control channel between the Control and Data planes is also vulnerable to attacks. The attacker can take control of the channel and can be ex-

ploited by the attacker to perform different attacks which include Eavesdropping, MITM, ARP Poisoning, ARP Flood, etc.

4. Attacks at Data Plane: The data plane is the plane that is responsible for just making the devices forward the data, making them simple. The separation of forwarding and data plane makes things simple but also prone to security. So, various attacks can occur in the Data Plane of SDN which include:

   (a) DoS Attack: It is an attack where a legitimate user is denied the services offered. This attack occurs at the Data Plane when a malicious user is sending a large number of packets to the switch in such a way that time between subsequent traffic flow is so less that the switch flow table becomes full and it cannot handle legitimate traffic [7, 8].

   (b) Traffic diversion attack: This type of attack occurs at the data plane where the switches divert traffic and original routes are not followed.

   (c) Side Channel Attack: This type of attack occurs when the malicious user tries to gather the timing information of the switch to connect to the controller [9]. In this attack, the attacker can get a hint of whether the flow rule for that flow exists or not.

   (d) ARP Poisoning attack: During the host discovery, the ARP table is consulted to check the IP/MAC pair. There are different ways by which the attacker can disturb the ARP table information and perform the attack [10]. The attack is performed by sending the malicious information to the ARP table of the genuine hosts so that all the traffic destined for the genuine host is sent to the attacker. ARP Poison is one of the ways to perform MITM attack, Denial of Service attack [11].

   (e) ARP Flooding attack is done to flood the ARP table of the host by random host machine's MAC addresses. The ARP table is flooded to its capacity by random hosts which leads to delay in the processing of genuine requests. The attacker can craft an ARP reply or ARP request packet and perform the ARP Flooding attack. There are various ways in which an attack can be prevented in a network. A secure link establishment can help overcome these attacks [12]. The proposed solution for preventing the attack is efficient as it does not require manual feature engineering and is so efficient in terms of time and processor load.

2.2 State-of-the-Art Research in the DDOS attack

A Denial of Service attack is an attack in which the goal of the attacker is to overflow the resources of the target host to interrupt the benign host. Existing solutions based on DDoS attack detection are based on either statistical techniques and others are based on applying machine learning techniques but they have not provided quite promising results. Some of the DDOS attack detection techniques are mentioned below:

Buragohain and Medhi [3] presented a statistical approach for the detection of DDOS attacks, by setting a constraint on the number of requests made per user. Features such as traffic rate and the total time of a flow duration in the switch are analyzed and their upper and lower limit are recorded for a genuine user. The algorithm analyzes these two features and classifies the traffic. Lower and upper value is defined after analyzing the usage pattern of a normal user. The set can be defined as a tuple: {Min_traffic_rate,Traffic_rate_max, time_min, time_max}. Incoming traffic has to follow the above bounds to be classified as benign traffic otherwise it will be considered malicious. The attack is prevented by reporting to the controller application the lower and upper limits. But this solution is a threshold-based solution which is not a scalable solution.

To address the threshold-based problem discussed above, Kalkan et al. [13] proposed performing Entropy calculation, along with TCP layer attributes such as Protocol type, TCP flags, Destination port, Source port, Destination IP, Source IP, and Packet size, for detecting the attack. It creates a pairing of the above-mentioned traffic features and calculates the joint entropy (for example, destination IP with TTL) when it is not attacked, then compares it to the same pair after the attack. If the difference in entropy values between normal and attack traffic is greater than a set value, the attack is detected, and the mitigation module begins to execute, in which detected pairs from the previous phase are compared to the same pairs from normal traffic flow, and a ratio is calculated. If the value of the ratio exceeds the set limit, it begins dropping packets for that pair until the bandwidth is reduced to acceptable levels. This approach deals with laborious entropy calculation tasks for a larger network.

To tackle the problem of laborious calculations, a novel solution is provided by Wang et al. [14] which applies the tensor-based method for DDOS attack detection. Tensors and Eigenvectors are collectively known as Eigen tensors. Tensors are multidimensional arrays where their constituents are defined as a tuple {a1, a2, a3, a4, . . . ,a_n}. Tensors are utilized with higher-order

data. The decomposition method is used to lower the dimensionality of the data. Tensors are represented in the graph in the form of vectors. Tensor multi-mode product is represented as

$$A_{\mathrm{M}} * X = \lambda * X \tag{2.1}$$

Here A is a tensor defined as $\mathrm{A} \in \mathrm{R}^{I_1,I_2,I_3,\ldots,I_m}$, X is the eigen tensor of A and λ is the eigenvalue of A. Data in SDN has been divided into three types namely basic data, parsed data, and instruction data. All the data can be divided based on their source like flow status information, and security information. The data-driven tensor framework for network attack detection uses local tensors to represent data (Source IP, destination IP, source port, destination port, time, number of packets, number of bytes, protocol, network topology data) in SDN. Through tensor operations data from heterogeneous sources are fused into a unified form. Flow table data is also represented as local tensors like matrices which are compressed and later sent to the controller. The tensor data of SDN is prepared by using three features namely Source IP, Destination IP, and Time. These features are used for the construction of benign and malicious vectors. A squared prediction error is computed and compared with the threshold parameter of Q-statistics. If the value of Q-statistics is greater than the threshold, the attack is detected.The proposed method uses a tensor, which is a complex solution to perform, and also used threshold value however, it is not made obvious how to select the threshold.

To deal with the complex solution issue, Researchers used novel Machine learning and Deep learning as DDoS attack detection techniques. One such solution is provided by Meti et al. [15] who proposed the use of various Machine-Learning algorithms to detect the DDOS attack in Software Defined Network (SDN).Various Machine learning algorithms are tested for their performance in the classification task. The dataset used is a TCP traffic set between certain locations obtained from the experimental results. The detection and prevention take place with the help of a controller application. The application keeps an Access Control List (ACL) with the help of which it segregates the TCP traffic set into normal or malicious traffic. This labeled traffic set is used by the machine learning classifiers. It is found that the SVM algorithm gives better results than other ML algorithms used. The machine learning algorithm involves manual feature extraction which is not an efficient method and can be improved by using deep learning algorithms. To tackle this problem of manual feature extraction in case of large datasets Liu et al. [16] proposed Deep learning models based on CNN & RNN for traffic classification into

the attack and benign classes. The dataset used for training and testing includes DARPA-1998, KDD-99, and others. The attack classes are DoS, R2L, U2R, and Probe. The deep learning methods of Deep Belief Network (DBN) and Deep Neural Network (DNN) are used for classification. There are different Data Preprocessing methods used by the author which include:
a) Unfiltered Byte Stream b) Filtered byte Stream c) Unfiltered word embedding d) Filtered word embedding e) Expanded Filtered word embedding. The first preprocessing method used raw data whereas the second method used ASCII code converted bytestream. Filtered word embedding involves the use of the dictionary to filter the payloads, which are then converted to word embedding vectors. Expanded Filtered word embedding means additional references are added to the dictionary, and the payloads are filtered and converted to word embedding vectors within the new dictionary. The accuracy was obtained by using Expanded Filtered word embedding. attained the highest accuracy of 94.11%. This method does not employ SDN specific dataset and thus cannot achieve a significant accuracy in attack detection. There are many solutions available for DDoS attack detection using ML technique but most of them utilize traditional dataset to train the ML algorithm. A significant accuracy can be obtained if the use of SDN-specific dataset can be done.

Another ML solution is provided by Latah and Toker [17] who investigated and evaluated an anomaly-based intrusion detection approach using NSL-KDD Dataset. It provide the results in terms of the following evaluation parameters: a) Accuracy b) Recall c) Precision d) False rate e) f1-measure f) Execution time g) AUC h) McNemar's test. Features related to content (i.e., F11-F22) are not used from the NSL-KDD dataset instead filtered dataset is used. The classification algorithm such as Extreme machine learning, SVM, Naive Bayes, Decision tree, Bagging Trees, LDA, neural networks, Random forest, AdaBoost, and K-nearest-neighbors are trained and tested on the dataset. When evaluated against accuracy, precision, AUC, and F1-measure, and McNemar's test Decision Tree achieves highest performance. The best results is given by AdaBoost. Even with the use of hybrid machine learning methods, the suggested approach does not yield improved results since it uses conventional datasets that were not produced in an SDN environment.

To resolve the issue of traditional dataset in DDoS attack detection, the effort has been made in this direction by Niyaz et al. [18]. The author proposed a novel method for DDOS attack detection using a stacked autoencoder (SAE). The different types of traffic collected include TCP, ICMP, and UDP traffic. Traffic headers are extracted at a regular interval of time. It checks

whether the packet flow is symmetric or not. If the flow is symmetric, then it is assumed as a legitimate flow else it is suspicious as an attacker. A total of 68 features is extracted from the network based on TCP, ICMP, or UDP traffic. For each batch of features, the median, entropy, and packet per-flow are calculated. Having followed the extraction of these features, the SAE classifies the traffic into benign and seven attack classes with an accuracy of 95.65%. It also performs two-class classification, assuming seven attack classes as one and then classifying traffic into two benign and attack classes with an accuracy score of 99.82%. The proposed solution provide significant results but the dataset has not been made available to the research community.

Another machine learning solution which utilizes SDN-specific dataset for attack detection on the control plane is performed by Anand et al. [19] who researched on the attacked control plane. In this paper, the author used nine significant features from OpenFlow traffic to detect five SDN attacks on the Control Plane. The attacks can slow down the network for genuine hosts and lead to DoS attacks. The author used six ML algorithms for the detection of attacks on the controller. This approach relies on the Open Flow traffic at the switches. The features which are extracted from the network are: 1) Participating capacity of switch 2) Switches participating in traffic 3) Number of nodes incidental 4) Count Index 5) Timeout Index 6) frequency of Drop Actions 7) Switch count 8) Packet in Packet Out Ratio 9) Packet in Packet Out Disparity. There are two types of traffic patterns that are generated:

- Ping all traffic: The traffic is generated by sending ping commands to all the hosts individually and then the dump is taken.
- Randomized traffic: With the help of Distributed Internet traffic generator DNS and VoIP traffic is simulated and its dump is taken.

The controllers which are used in the network are arbitrarily picked from either of these: (i) Normal, (ii) Attacked Id-1, (iii) Attacked Id-2, and (iv) Attacked Id-3. For evaluating the significance of each feature plotting a sample of data along with their classes is done. The class indicates the category of the controller, either attacked or normal. The classifiers which are used for detection of the attacked controller are (i) Naive Bayes (ii) Adaboost (iii) Multilayer Perceptron (iv) Support Vector Machines, (v) Random Forest (vi) Decision Tree. Amongst which random forest achieved the highest accuracy of 97% in detecting the compromised controller. The

proposed solution work on SDN-specific features but does not make the dataset available to the research community. The method is identical to the one we proposed to detect DDOS attacks, but we also produced the SDN-specific dataset and made it available to the research community.

Table 2.1: State-of-the-art in DDoS attack in SDN.

S.No	Year	Author	Main Contribution
1.	2015	Dabbagh et al.	Description of SDN architecture and challenges encountered in dealing with various security attacks including DoS, DDoS attack, Side channel attack.
2.	2019	Liu et al.	Utilized Deep learning models based on CNN & RNN for traffic classification into the attack and benign classes.
3.	2018	Anand et al.	Identified five threat vectors which can be detected by the machine learning algorithm.
4.	2018	Latah and Toker	Uses machine learning based approaches to predict the attack.
5.	2016	Buragohain and Medhi	Used various traffic features to detect the DDOS attack and categorise the traffic.
6.	2015	Wang et al.	Used the tensor based method for DDOS attack detection. Tensors are multidimensional arrays which uses three features namely Source IP, Destination IP and Time.
7.	2016	Da Silva et al.	Used the concept of machine learning to detect DDoS attack in which Support-Vector classifier achieved the highest results.
8.	2018	Kalkan et al.	It used the statistical solution to detect DDOS attack in SDN. The attack is detected by calculating the difference between the Joint Entropy of attack traffic and normal traffic.
9.	2017	Meti et al.	Discussed the use of machine learning techniques such as Nave Bayes and Support Vector Machines to classify traffic flows as benign or malicious.

Continued on next page

Table 2.1: State-of-the-art in DDoS attack in SDN. (Continued)

S.No	Year	Author	Main Contribution
10.	2017	VinayaKumar et al.	With the KDD-Cup-99 Dataset, we proposed the use of CNN-LSTM and other hybrid models for network intrusion.
11.	2016	Niyaz et al.	A deep learning method of Stacked Auto-Encoder for detecting DDOS attacks has been proposed. There are a total of 68 features extracted from the traffic headers, and the traffic is classified as normal or attack.
12.	2020	Abdulla et al.	A cloud- based SDN Architecture has been proposed to prevent flow table attack, control plane attack, and Byzantine attack by the use of certain policies.

2.3 State-of-the-Art Research in ARP-Based Attacks

ARP-based attacks that occur in the traditional networks also affect the hosts and other nodes in SDN. The ARP vulnerabilities were also present in the traditional network but SDN handles them differently.

Existing solutions are based on either checking the traffic against the stored binding which is a complex task in the case of a large network, or checking the pattern of traffic which is time taking task [20], cryptographic solutions is a complex task in terms of processing power or statistical techniques is also a computationally intensive task. To understand the topology attacks e taxonomy of topology attacks is shown in Figure 2.2.

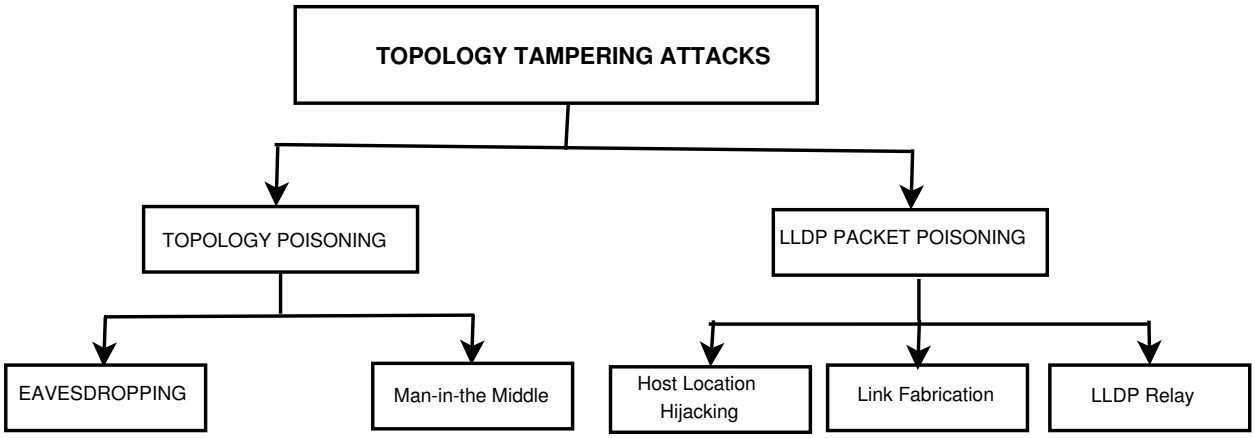

Figure 2.2: Taxonomy of Topology attacks in SDN

The following are some of the most recent State-of-the-Art research in SDN for dealing with ARP-based attacks:

Conti et al. [21] provided a extensive survey on Man-in-the-middle (MITM) attack. The survey includes a deep understanding of the attack along with the attacked security principles of Confidentiality and Integrity. The extent to which a MITM attack can affect the network traffic transmission is also discussed. Extensive classification of MITM attack is done based on various parameters like the location of attacker, nature of communication channel, and impersonation technique. The author clearly explained the background idea required to comprehend topology-based attacks. A work to detect MITM attacks was proposed by Hong et al. [22]. The author tried to collect the dynamic information of the links in the topology which helps to detect the attack. The information collected includes a data structure that records the name and number of switches through which a particular flow passed. The data structure generated includes Source and Destination IP address, Source and Destination MAC address, and sequence of ports through which the flow passes are used to detect the attack. The detection mechanism detects the position information of the attacker and breaks the attack by passing the updated flow rules to the Controller. But this solution seems to be a static solution that will compromise the programming capability of SDN.

To overcome the above problem, Nam et al. [23] proposed MITM-Resistant ARP which is a modified version of the existing ARP protocol to protect against the ARP poison-based MITM attack. The author proposed to store the original IP/MAC pair in a table for an hour and as long as the host in the network is present, it updates the original IP/MAC pairs by repeatedly sending the ARP request. If no ARP reply is received within the threshold time, the particular IP/MAC

pair is disproved and deleted from the table. So, no attacker can poison the ARP tables as each host IP/ MAC pairing is preserved. However, there is a scalability issue with this approach in big networks, making it difficult to keep such a vast list of IP/MAC pairs.

Carnut and Gondim [24] proposed an algorithm for calculating the count of ARP requests and replies sent over the network. If the count of ARP reply packets is greater than the count of ARP request packets then the Poison attack is detected. The disadvantage of this method is that collecting and analyzing the statistics can be a very computationally intensive task.

To overcome the above problem, AbdelSalam et al. [25] detect the ARP Poison attack by comparing the Source MAC address in the Ethernet frame and ARP header which if different detects the presence of spoofed ARP request packets in the network. Similarly to detect the spoofed ARP reply packet, the Destination MAC address in the Ethernet frame and ARP header are compared. In the situation of a mobile host, the static method of packet inspection will not be a solution.

To avoid the above issue, Deng et al. [26] detect the controller attack by ensuring the legitimacy of Packet-In messages. So the defense mechanism which has been followed is to maintain a MAC to port mapping table. The table is comprised of all the host mac addresses and port numbers of the switch to which they are connected when the topology starts. So whenever a new packet_in reaches the controller it matches the MAC address in the packet_in with this table MAC address, if the MAC address exists in the table, then it processes the packet else it drops it. This approach is vulnerable to internal attacks; if an inside host commits an attack, the proposed solution will not work.

To solve the above issue Zhang and Qiu [27] detect the MITM attack by calculating the packet delays of a particular TCP connection. It then compares the mean of the delays of a particular session with the old referenced values. It was found that when the delay is more than the predefined delay value, the suspicious outlier is informed to the monitoring module and an attack is detected. The proposed technique is not scalable and does not address dynamic real-world situations. Only using delay values to identify attacks may result in inaccurate predictions.

To address the scalability issue Hoz et al. [28] worked on MITM attack detection by enhancing the security of the transport layer. According to the author, Transport layer security (TLS) is based on secure key exchange, an insecure key exchange can lead to Man-in-the-middle attack. Securing the key exchange process involves securing the certificates use for key exchange. Various work has been done to detect the untrusted certificate but all of them suffer from some limi-

tations. It detected untrusted certificates by collecting information from different sources. However, this technique is constrained by extensive cryptographic operations including challenging key exchange.

To resolve this problem of complex cryptograhic operations Nehra et al. [20] provided pattern based solution. The attack is detected by sending a malicious ARP request from the attacker. In normal communication, every ARP request is followed by an ARP reply and IP packets but in the case of ARP Poison attack and ARP Flooding attack there is no consecutive IP packet, but just the ARP packets that indicate the attack. The detection module whenever finds this pattern, an ARP Poison attack is detected. This approach for detecting the attack is unique as it is based on finding a particular traffic pattern as described above, which is an elegant way of detecting the attack. Previous work tends to use either cryptographic techniques or statistical techniques which are time-consuming and complex. The proposed approach in our work is similar but the authors have logged the various features and created a dataset. In the proposed work, the application of machine learning algorithms has been done to detect the attack which makes it different from this work.

Wang et al. [29] proposed a novel architecture for detecting MITM attacks. The machine learning model automatically learns network traffic features by using a CNN for learning the spatial features, followed by LSTM layers to learn temporal features. The developed model is trained on Darpa and ISCX-12 dataset to validate the results. So, the paper used the combination of CNN and LSTM to achieve better results. Because the features are learned automatically, it has reduced the False Alarm Rate. The proposed method made use of a traditional dataset that lacked SDN-specific properties, making it incapable of accurately identifying attacks in an SDN environment. As a result, the thesis generates a dataset specifically for SDNs and makes it accessible to the research community.

Table 2.2: State-of-the-art in ARP-Based attacks in SDN.

S.No	Year	Author	Main Contribution
1.	2016	Conti et al.	Provides a comprehensive survey of MITM attacks, including a thorough understanding of the attack sources, the specific security principle being attacked, and the scope of impact.
2.	2010	Nam et al.	Proposed MITM resistant ARP by employing a threshold-based approach which does not allow the attacker to operate in the network.
3.	2017	Nehra et al.	Proposed a pattern matching method for detecting the ARP Poison and ARP flood attack.
4.	2019	Sebbar et al.	Proposed a threshold-based approach using the state of the node and the duration of connection for detecting the MITM and traffic redirection attack.
5.	2018	Carnut and Gondim	A statistical method for comparing and calculating the number of ARP requests and ARP replies has been proposed for detecting ARP Poison attacks.
6.	2014	Hoz et al.	A solution for MITM attack detection was proposed by providing secure key exchange in the transport layer.
7.	2019	Chou et al.	Proposed solution for topology injection, flooding attack by finding the correlation between the various links of the topology.
8.	2016	Wang et al.	Used CNN and LSTM models on DARPA and ISCX-12 datasets to detect MITM attack.
9.	2018	Zhang and Qiu	Used the statistical method of MITM attack detection by calculating and comparing the packet delays of a TCP connection with mean of the delays.

Continued on next page

Table 2.2: State-of-the-art in ARP-Based attacks in SDN. (Continued)

S.No	Year	Author	Main Contribution
10.	2017	Deng et al.	Proposed solution for detecting the controller attack by checking the legitimacy of Packet_in messages with the help of MAC to Port table.
11.	2015	AbdelSalam et al.	Proposed method for detecting ARP Poison attacks by comparing the Source MAC address in an Ethernet frame to the ARP header.
12.	2015	Hong et al.	Proposed solution for MITM attack detection by finding out the position of attacker and thus updating the flow rules to deny the packets from attacker to flow in the network.

2.4 Preliminary Work

In this section, we will present preliminary work for DDOS attack detection using a threshold-based approach. Threshold-based approaches are presented by various authors, but this approach is different from them as we have emulated [30] the network in SDN.

In Distributed Denial of Service Attack (DDOS), the adversary's aim is to saturate the target host with so many packets that it lets suffer the legitimate user in using the network services. Adversary performs the attack in such a way that it leads to both data and control plane saturation. Adversary performs the following attacks:

- Flooding attack: Adversary performs the flooding attack to compromise the bandwidth and switch memory and thus controller delays access to benign users. To detect such an attack we have proposed a threshold-based approach which is explained in the next section.

- TCP-SYN Attack: In this attack, the adversary sends a TCP Syn Packet to the target but the destination host is yet to send the ACK to the source in the meantime adversary sends another Syn Packet for connection establishment, so it becomes difficult for the target to handle the traffic. This is because the controller already allocates all its resources for the

previous Syn requests and so the controller reaches its saturation to handle the packets

2.4.1 Methodology

We proposed a countermeasure known as Continuous watch. It detects the attack by continuously analyzing the traffic statistics on the switch. Detection and prevention of the attack is done in two phases. In the first phase, the attack is detected by analyzing the various statistical parameters listed in Table 2.3. The attack is prevented in the second phase by modifying the routing concept of the target host in the switch flow table to drop the packets.

Table 2.3: Flow statistics extracted from SDN for DDOS attack detection

S.No	*Flow statistics*	*Meaning*
1	byte_count	Count of bytes
2	duration_n_secs	flow duration in seconds
3	priority	Priority of the flow
4	hard time out	set time when flow is removed
5	idle timeout	set idle time when flow is removed
6	len	length of the message
7	match	can be IP, Mac address, Port No.
8	packet-count	it is the total count of packets

1) Primary phase of detection: In this phase, a thread is run continuously to monitor the switch. At regular intervals of time, it collects the flow statistics which include various parameters that are mentioned in Tables 2.3 & 2.4. Packet count is used in the work to calculate the packet rate. The packet rate value is used to check for the attack, whenever the packet rate is greater than a predefined threshold value of 100, the attack is detected and appropriate action is taken. Algorithm 1 shows the pseudo-code for the detection of the attack.

Table 2.4: Port statistics extracted from SDN for DDOS attack detection

S.No	Flow statistics	Meaning
1	txbytes	Bytes sent on a port in network
2	rxbytes	Bytes received on a port in network
3	txpackets	Packets sent on a port in network
4	rxpackets	Packets received on a port in network
5	txerrors	Errors during sending the packets
6	rxerrors	Errors during receiving of the packets
7	port id	Port number
8	datapath	switch id
9	duration	time during transmission
10	in_port	entry port
11	out_port	exit port

Algorithm 1 Proposed Algorithm for detection of DDOS attack in SDN

Input: Traffic statistics at switches
Output: Attack detected successfully

Initialisation:*packetratelimit*=100
For each flow collect the *packet count* and textittx_ & rx_ byte statistics
for $i = packet1$ to $packet\_n$ **do**
    textitpacketrate=packet_count/duration
    textitBandwidth=((tx_bytes+rx_bytes)*8)/1024
    if (packetrate(i) $\geq packetratelimit$) **then**
        Attack detected
    end if
end for
return Flow with $SourceIP$, $DestinationIP$

Secondly, we also compute the bandwidth for every 30 seconds. Bandwidth is computed for all the ports by specifying the OFPP_ANY to the OFPFlowStatsRequest [31], [32] to measure the bandwidth of all the ports. We can specify the specific port number if we want to measure it from a particular port. Bandwidth is also found to be increasing high when the attack takes place, but once the prevention technique is used then bandwidth will be reduced.

Algorithm 2 Proposed Algorithm for prevention of DDOS attack in SDN

Input: Flow for the Packet Rate, Bandwidth
Output: Deleted BlockedList

For the flow which has the identified packet rate and bandwidth
for $i = flow_1$ to $flow_n$ **do**
    append the corresponding source IP of that flow into blocked list which is a list of IPs who have exceeded the threshold packet rate.
    if ($SourceIP$(i)== $BlockedList$(i) **then**
        Modify the action of that flow to drop the packets from source IP identified.
    end if
end for
return Deleted $BlockedList$

Secondary phase & Countermeasures

In this phase, Attack prevention takes place. The attack which is detected during the primary phase is prevented by changing the forwarding logic of the switches by modifying the flow rule in the flow table to drop the packets by using OFPFlowMod message instead of forwarding. Algorithm 2 shows, the pseudo-code for the prevention of the attack.

2.4.2 Experimental Work and Results

We used Mininet [33] as an emulator to analyze the network traffic, the attacks, and their countermeasures along with Ryu [34] as a network controller. The controller is written entirely in python and we have deployed our application logic as an application on Ryu.

Various simulation parameters are shown in Table 4.3. To analyze the results we have plotted the graphs of certain parameters without prevention technique and with prevention technique which will help us understand the scenario of attack in a better way.

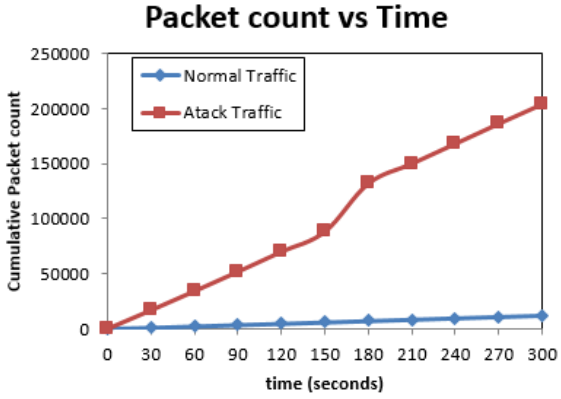

Figure 2.3: Impact of attack on the Packet count Parameter

Table 2.5: Simulation Environment to check the proposed approach of DDOS attack detection.

S.No	Parameters	Value
1	Host OS	Windows10
2	Guest OS	Ubuntu16.04
3	VirtualBox	5.1.26
4	Emulator	Mininet
5	Controller	Ryu
6	# Controller	1
7	# Switches	9
8	# Hosts	12
9	Protocol Used	OpenFlow
10	Graphical package	MiniEdit
11	Traffic Generation tool	Iperf, Curl, Ping
12	Controller Port Number	6653
13	Bandwidth	100 kbps
14	Simulation Time	300 seconds
15	Packet rate threshold used	100 packet per second
16	Statistics collection interval	30 seconds
17	Bandwidth plot interval	30 seconds

Figure 2.3 shows the effect of the attack on the number of packets. As can be seen from the Figure that when there is no attack the number of packets over the network is in the range of (1000-12000) packets when measured after every 30 seconds. Similarly, when a malicious node attacks the target host (H1) the packet count shows an abrupt increase and reaches the range of (18000-200000) packets. It is a strong indication of the attack. The increase in packet count is due to the malicious nodes(H2, H3, H4) attacking the target host (H1) at 100 pps by each of the malicious nodes. Host 2 will send on port 2, so there will be 3000 packets every 30 seconds and on port 3 there will be 6000 packets every 30 seconds. So three malicious nodes attack the target host and packet count increases in the interval of 9000 packets every 30 seconds and a total of 18000 as twice count for request and response.

For preventing the attack we have calculated the packet rate, which is defined by the number of packets sent per second. To check the attack, packet rate threshold value of '100' is used and if the packet rate at a particular switch crosses the threshold rate then the attack is detected and the control goes to the prevention module. The value of '100' for packet rate is used because when the normal traffic flows the packet rate values lie between 5-8. But during attack, only the packet rate lies above 70 so we have chosen a value '100' which occurs only when there is

an attack. Also, Packetcount of '100' can be from a legitimate user but a packet rate of '100' cannot be from a legitimate user as confirmed experimentally.

The prevention module alters the host's routing concept to prevent further forwarding of packets by modifying the action part of the flow table to decline packets instead of forward, preventing the malicious source IP from sending packets again.

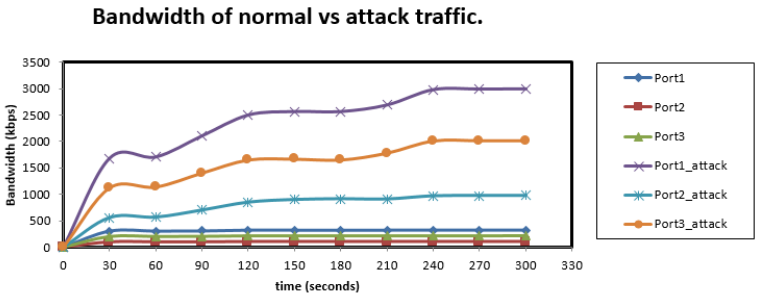

Figure 2.4: Bandwidth during Normal vs Attack traffic.

Figure 2.4 depicts the bandwidth graph plotted between normal and attacked scenario. In the graph, we can infer that bandwidth of the network during normal traffic ranges between (100-300 kbps) which shoots up during the attack in the range of (1600-3000 kbps) which is an indication of an attack. Port1 in yellow is the port that is connected to host1 which is the target host and it is the sum of bandwidth at host 2, host 3, and host 4 who have attacked the target host at 100 pps.

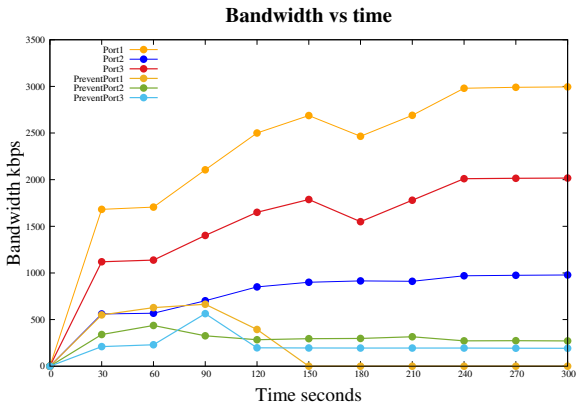

Figure 2.5: Traffic Bandwidth in attacked vs after countermeasure applied

Figure 2.5 depicts the bandwidth of the network when countermeasure is applied which stays in the range of (400-600 kbps). It is because of the packet rate threshold which has been set to 100 packets per second, which when the attacker (H2, H3, H4) continues sending packets, they

are dropped and port1 drop the packets and its bandwidth falls to zero. There is packet sending still done by Host 2,3,4 but Host 1 drop the packets can be seen from Figure 5 that port2, 3, 4 have not fall to zero.

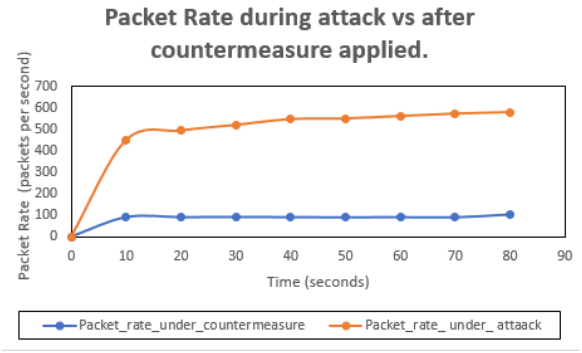

Figure 2.6: Packet Rate during attack vs after prevention applied.

Figure 2.6 depicts the graph between packet rate before and after the countermeasure is applied. When the malicious hosts(H2, H3, H4) attack the target host (H1) attacking at 100 pps the attack rate shoots up in the range of (600-744) but after applying the countermeasure whenever the packet rate goes above 100 the packets are dropped which is shown in the Figure that when the packet rate becomes 106 the switch started dropping packets and so packet rate comes down.

2.5 Research Gap and Problem Statement

SDN security is critical since the centralized architecture is vulnerable to network threats that can bring the entire network down. Aside from typical threats, there are new attacks that can occur in SDN that must also be addressed. The problem of identification of attacks in SDN using machine learning techniques has been investigated in the thesis. There has been a lot of research done in the past by numerous researchers, however, the majority of them have flaws. The following are some of the research gaps that have been identified:

- Previous research work [15]detect the attack by using a threshold-based approach, which is not feasible in a large network,
- Most of the work used NSL-KDD Dataset [25] in detecting the attack, which is not an SDN-specific dataset.

- Most of the work [35] does not provide details about the features used for detecting the attack.
- Most of the work just explained the architecture [36] without explaining the details of the experimental work.
- It also does not provide any experimental results to validate the theory.

2.6 Objective

The thesis aims to work on DDOS and ARP-based attacks which occurs in SDN. For detection of these attacks, the datasets have also been generated. Machine learning algorithms are trained on the generated datasets and attack detection and traffic classification into benign and attack classes have been attained. To achieve these goals, we've taken the following steps:

- Generated SDN datasets for attack detection on the Mininet emulator, which will include extracted traffic features to distinguish attack traffic from genuine traffic.
- Designed an efficient algorithm for detecting network attacks in SDN.
- Performed Comparative analysis of variants of proposed attack detection classifier in terms of classification accuracy, training time, confusion matrix, and F1-score.
- Developed an algorithm for mitigation and prevention of security attacks in SDN and provide a comparative analysis of the proposed attack prevention algorithm with a traditional prevention algorithm.

CHAPTER 3
AUTOMATED DDOS ATTACK DETECTION IN SDN

The previous chapter discussed existing research against different security attacks. The techniques are primarily based on calculating the entropy [37], calculating statistical parameters [38] such as standard means of deviation, static IP/MAC mapping, and cryptographic computations to detect the attack. This work provides a one-of-a-kind solution for DDoS attack detection, that includes the generation of a DDoS attack dataset. Attack detection has been performed using the Machine learning algorithm on the dataset which provides significant traffic classification results [39].

3.1 Introduction

Software-Defined Networking (SDN) is a communication design that is defined by a software program. It is a centralized architecture where the network is controlled by Controller. Examples of SDN controllers include FloodLight, Ryu [34], Pox, OpenDayLight [40], Nox, etc. which are open source and incorporate a set of APIs for building network applications.

But the benefits offered by SDN come at an expense of security. The centralized architecture is prone to security attacks different planes [41,42] of SDN architecture. The proposed research focuses on the classification of network traffic into benign and DDoS attack class by applying ML Algorithm [43].

In the DDoS attack [44,45] presented in Figure 4.2, the attacker's goal is to fill up the resources of the target host to disrupt the benign host. DDoS attacks in SDN can occur at different architectural planes. So far, several authors [2,18,46–49] have worked for DDoS attack detection in SDN. But these works mostly used unrealistic topologies. Others have used traditional datasets for detecting the attack on SDN. However, some have used the SDN testbed but have not made the data public to validate the approach. The cause for the inapplicability of traditional methods in SDN is the architectural difference between the two networking paradigms.

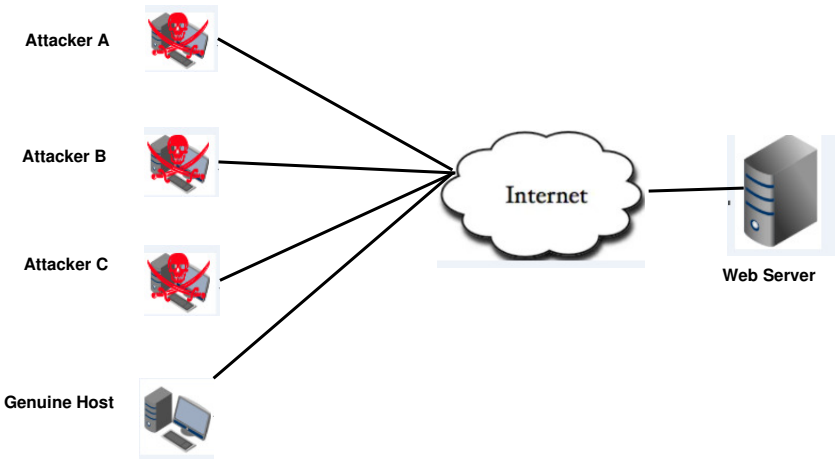

Figure 3.1: A scenario of DDoS attack by flooding the target resources.

This motivates the author to work on the SDN testbed and create the SDN traffic Dataset.

3.1.1 State-of-the-art VS Proposed Method

At present, the work on DDoS attack detection in SDN is already addressed. But the selection of the significant features which play an important role in attack detection is done in the proposed method. Besides, a hybrid method of Random-forest with Support-vector-classifier is used for the classification task in the proposed work. Several significant works for DDoS detection have been done using Machine learning and deep learning [50]. However, the proposed work differs significantly from them in the following ways.

Francesco Palmieri [46], latah and Toker [51], Wang et al. [14] have worked on DDoS attack detection but did not use the SDN emulated traffic dataset. Indeed they used a publicly available dataset that has been generated for traditional network architecture and has different set of features that is not applicable in SDN environment. Da Silva et al. [2], Buragohain and Medhi [3] have worked on the DDoS attack detection using machine learning approaches where they have considered just two features which can not justify the accuracy achieved whereas the proposed method used twenty-three features.

Niyaz et al. [18] worked on the DDoS attack detection using the Deep Learning technique but the paper has a few limitations in terms of processing capabilities as it has considered 67 features. We have to optimize the number of features to make the system efficient. The proposed work has used 23 features and attained higher accuracy. Santos et al. [47] have attempted to work on their dataset but they do not follow the real scenario as the dataset is prepared by labeling the traffic serially which means normal traffic followed by attack traffic. But in real-time, the

attack happens in between the normal traffic and the created dataset reflects the same scenario.

Myint et al. [48], Cui et al. [49] have claimed to work on their datasets but have not made them public for verification but in our work, we have shown the results tested on our dataset and made it public as well. None of the research works cited above used the hybrid model of SVC-RF which is used in the proposed approach and attained significant results.

The proposed approach in this work is composed of two modules. The first module collects the flow and port statistics to create the dataset and the second module applies a machine-learning algorithm to classify the traffic. The contribution of the work can be summarized as under:

- **Creation of SDN dataset using mininet emulator**: In the work, the authors have created the SDN traffic dataset [52] using mininet emulator [33]. The datasets which are already available like NSL-KDD, KDD-cup99, and ISCX are created for the traditional network and only contain a subset of features available in SDN. So, the author has created the SDN dataset generated in the mininet emulator. The dataset has been shared in the Mendeley data repository given in footer [1].

- **Classification of traffic into benign and malicious using Machine Learning**: After we create the SDN traffic dataset, different Machine Learning algorithms [53] are applied to classify the traffic into benign and malicious classes. This trained model that classifies the traffic, can also be used in real-time. As the dataset is created on the SDN emulator, the algorithm which produces the highest classification accuracy will also produce similar results in the real-time scenario.

- **Detection of the attack on the host**: To detect the attack on hosts, the effective features in the dataset are analyzed by the machine learning model and detect the type of traffic. The rate of normal and attack traffic is kept similar which made the problem of classification more difficult. But it also proves the validity of the features analyzed.

3.2 Dataset

In this section, the significant features generated during the dataset creation along with the System model and proposed methodology are discussed. The dataset generated during the first phase is used in the second phase for traffic classification.

[1] https://data.mendeley.com/datasets/jxpfjc64kr/1

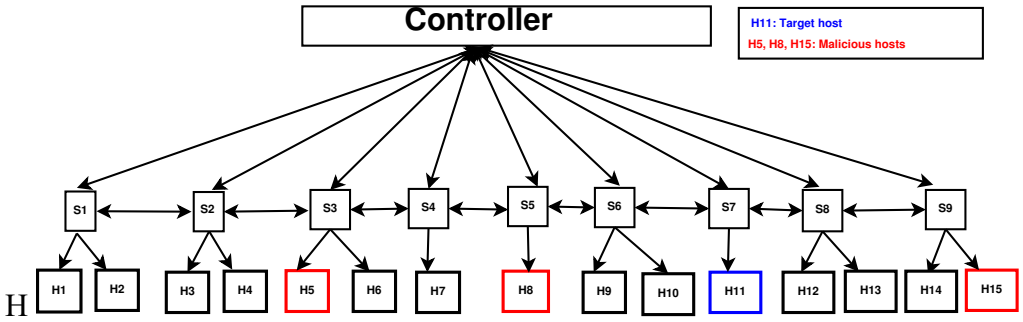

Figure 3.2: Topology utilized for dataset creation.

3.2.1 Dataset Creation

Dataset creation is done by extending the controller with a python application which is created with the help of RYU API [54]. It monitors the switches present in the topology and collects the various flow and port statistics at a regular monitoring interval. It also writes the collected statistics into a CSV file. Sample topologies used is shown in Figure 3.2 which shows hosts connected with different benign and malicious hosts in the topology.

As the dataset is emulated, it does not contain any missing data. The total number of records is 1,04,345, and each record consist of 23 features, and some are shown in Table 3.1.

Table 3.1: Features Used in the Dataset

S.No	*Features Used*	S.No	*Features Used*
1	Packet Count	9	Source IP.
2	Byte count.	10	Destination IP.
3	Total number of flows in a switch.	11	Number of Packet_in messages.
4	Packet Count per-flow.	12	Packet Rate.
5	Byte Count per-flow.	13	Port number.
6	Duration_n_secs.	14	txbytes.
7	Duration_ secs.	15	rxbytes.
8	Total_duration.	16	Port bandwidth.

In the dataset, the time field represents the date and time at which the data is recorded, switch represents the datapath-id in the topology, src represents the Source IP Address, DST represents the Destination IP Address, Pktcount represents the count of packets sent during a flow, byte count represents the count of bytes sent during a flow, duration represents the time during which the flow remains in the switch, dur_nsec represents the duration in nanoseconds during which the flow remains in switch and total duration is the total sum of dur_sec and dur_nsec, Packet_in

represents the count of Packet_in messages conveyed to the controller, port_no represents the port number of the switch to which the requests are sent, tx_bytes represent the count of bytes sent on a specified switch port, rx_bytes represent the number of bytes transferred to a switch port, tx_kbps represent the kilobytes transferred per second, rx_kbps represent the kilobytes received per second and tot_kbps represent the bandwidth of a switch port. Dataset has been freely available for the research community [2]

3.2.2 Dataset Annotation

The annotation of the dataset is done automatically by applying some code logic. The coding is done in such a way that when benign traffic runs, the label column of the dataset is set as ”0” and for malicious traffic, label is set to ”1”. After annotation of the data, we apply a particular ML algorithm to classify the traffic. The two classes of traffic are a) Benign b) Malicious which correspond to 0, 1 respectively. The total count of traffic instances present in the dataset is depicted in Table 3.2.

3.2.3 Feature Description

This section discusses the significant features present in the dataset and utilized by a machine-learning algorithm to classify the traffic effectively. The various features are critically analyzed and explained below.

- Average Packet count per flow (APPF): In SDN, an attacker take advantage of the fact that source IP is used as a distinguishing feature for identifying a flow. So, the attacker utilized spoofed IP addresses and flood the flow table with different IP addresses. The attacker only aims to flood the flow table and not send data packets. So, the average packet count per flow is a significant feature. APPF decreases in case of attack and increases in case of the benign user as the number of packets sent in attack is less. APPF is the fraction where the numerator is the packet count during a flow and the denominator is the count of flow entries present in the switch at a particular time. APPF can be expressed as under:

$$APPF = \Sigma_{i=I}^{i=n} c_i / f \tag{3.1}$$

Here, c_i is the Packet count of $flow_i$ (which is the total packet sent during the flow i) and

[2]https://data.mendeley.com/datasets/jxpfjc64kr/1

i ranges from flow$_i$ to flow$_n$. This feature can be used to detect DDoS attack.

- Average Byte count per flow (ABPF): In SDN, the average byte count per flow can also be used to indicate the attack. This parameter is calculated from one of the flow statistics i.e., byte count. It is calculated in the same way as for APPF. ABPF also decreases in case of attack. ABPF is the fraction where the numerator is the byte count during a flow and the denominator is the number of flow entries present in the switch. ABPF can be expressed as below:

$$ABPF = \Sigma_{i=1}^{i=n} b_i / f \tag{3.2}$$

  Here, b is the total bytes during a flow. This feature can also be used to detect DDoS attack.

- Total number of flows in a switch: A flow is defined as the transmission of packets between sending and receiving host in the network. A flow table is maintained at the switch that stores all the flow details for the topology. The total number of flow can be expressed as below:

$$f = length(flowtable) \tag{3.3}$$

  Here, flow_table represents the table maintained at the switch whose length gives the number of flow description at a particular instant of time. In case of attack, the switch connected to the target host is found to contain the maximum number of flow entries [55]. The periodic monitoring of the hosts shows that the target switch has a maximum number of flow entries. So, the number of flow entries is an important feature to consider for classifying the traffic as benign or malicious.

- Protocol: Protocol defines the protocol associated with the traffic flow. Protocol can help identify the traffic protocol associated with the malicious traffic. Any deviation from the normal traffic protocol can help detect the attack.

- Duration: Duration depicts the duration of flow entry in the flow table of a switch. Duration is computed by the sum of d and (d_$nano$). The total duration during which the flow remains in the switch's flow table is a sum of duration_sec (d_$nano$) converted into nanoseconds and duration_sec (d). Total duration in nanoseconds can be expressed as

below:

$$\textit{Total\_duration} = ((d * 10^9) + (d\_nano)) \tag{3.4}$$

Attack traffic remains for larger duration as compared to benign traffic. So, the duration during which a flow remains active in the switch plays an important factor in deciding the malicious host. Duration has a high value in case of attack as compared to benign traffic.

- Number of Packet_in messages: Packet_in messages are the messages processed by the Controller and sent by hosts when they lack a flow entry in the flow table and are thus unsure what to do with the received packet In response, the controller sends the Packet_Out message. When the attackers used spoofed IP addresses, a large number of Packet_in messages are reported to the controller. So, the count of Packet_in message is an indicator of an attack. When the Packet_in message is created by the switch it is known as an event. Event handlers are associated with every event generated. So, to obtain the number of Packet_in messages generated, counter is updated in the Packet_in message event handler routine. Count of Packet_in messages increases in case of an attack. So, it can be used as a significant feature for attack detection. The threshold above 1970 in case of UDP traffic, 4300 in case of TCP traffic, and above 3000 in case of ICMP traffic is considered as attack traffic in the dataset. When the number of Packet_in messages increases, there are generally a large amounts of packets sent by the attacker to the destination. So, the other features like average packet per flow and packet rate also play an important role while labeling the host as attacker or normal.

- Packet rate: Packet rate is defined as the number of packets sent per second. Benign and malicious traffic, both send traffic at the rate of 450 packets per second but as the attacker used spoofed IP addresses, decreases in the packet rate is reported. It can be expressed as below:

$$\textit{Packet\_Rate} = (p_{t+1} - p_t)/i \tag{3.5}$$

where p_{t+1} is the number of packets sent in a flow at time t+1 and p_t is the number of packets sent at time t and i is the monitoring interval. As the packet count is a cumulative number, so packet rate is calculated by subtracting the consecutive packet count and dividing by i.

- Port Bandwidth: Port bandwidth is defined as the sum of received bytes (r) and transmitted

bytes (t). The port statistics are collected from the switch's port at a regular interval. As the attacker sends more requests and less data, so the Port bandwidth is less which indicates the attack. Port bandwidth is expressed as below:

$$Bandwidth = (t * 8)/1000 + (r * 8)/1000 \tag{3.6}$$

Here t and r are the extracted Port statistics from the switch. The Port bandwidth is calculated by using them and specified in kbps.

3.2.4 System Model

In this section, we are going to describe the SDN architecture where the network is modeled as a directed graph G ∈ (V, E), where V is the set of nodes and E is the set of connections that connects various nodes including hosts, switches, and controller in the network.

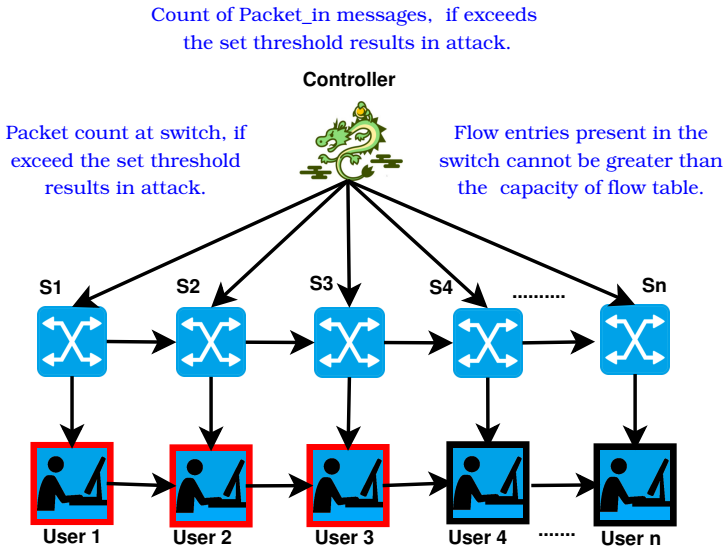

Figure 3.3: System Model Satisfying the Constraints

The problem statement is described by the following attributes:

- Input Data- The input dataset contains seven tuples: {C, S, t, θ, β, ζ, δ} where C is the constraint set that is to be satisfied by the network. S is the set of switches {S_1, S_2, S_3,..., S_n}, t is the time during which statistics are collected, θ & β are the thresholds used, above which the attack is said to take place, ζ is the maximum capacity of emulator and δ is the change in the value of various statistics collected, whose value can be analyzed to detect the attack.

- Constraint set (C)- is specified as four tuples {C_1, C_2, C_3, C_4} where each constraint can be explained as under:

  - $C1 : 0 <= P < \theta$

    Constraint C1 states that the Packet count at each switch should not exceed the threshold value θ. If the packet count (P) exceeds a set threshold value then an attack is said to take place.

  - $C2 : 0 <= P_{in} < \beta$

    Constraint C2 states that the count of Packet_In messages (P_{in}) to the controller should be greater than zero and less than a threshold value β. If the number of Packet_In messages exceeds the threshold limit set then the attack is said to take place.

  - $C3 : D \in$ 0,1

    Constraint C3 states that decision variable D can take only two values 0, 1 which correspond to benign and malicious traffic respectively.

  - $C4 : \Sigma_{i=1}^{n} \mathrm{F} \leq \zeta$

    Constraint C4 states that the number of flow entries (F) present in the switch should not be more than the processing capacity of the emulator(ζ).

    The System model can be explained in Figure 3.3. It depicts that the specified constraints are satisfied by different hosts. The various threshold values in the constraints, model the traffic behavior. If the traffic remains within the threshold value, the traffic is recognized as normal and if it exceeds the threshold value the traffic is recognized as malicious. We have generated the normal and attack traffic at a fixed rate of 450 packets per second. So the threshold of Packet count above this value can be set to check the data behavior. Similarly, in the case of attack traffic the count of the Packet_in messages exceeds the count for the normal traffic. In an attack scenario, a large number of requests are made from spoofed IP addresses which results in a larger number of Packet_in requests to the controller. Also, D is a decision variable that takes either of the two values of 0 or 1. All these constraints are set from the analysis of the dataset.

3.3 Proposed Methodology

This section discusses our proposed methodology. The proposed algorithm 3 shows the step-by-step process of dataset creation. The algorithm collects the flow statistics from the switches present in the topology. For every event there is an event request and reply handler, so during the monitoring interval flow statistics are retrieved by calling OFPFlowStatsRequest method. In reply the event reply handler i.e., OFPFlowStatsReply method is evoked which returns the flow statistics. Similarly, Port statistics are retrieved by calling OFPPortStatsRequest method. These statistics are written to the CSV file during the monitoring interval for all the topologies and the dataset is created.

Algorithm 3 Proposed Algorithm for creation of DDoS attack Dataset in SDN

Input: Traffic statistics at switches. Flow and Port statistics are extracted from the switches at a regular interval of 30 seconds.

Output: SDN traffic Dataset after appending all the flow and port statistics.

Initialization:*Normal traffic Packet per flow, Attack traffic Packet per flow, Packet_in count, flows count, Packet Rate.*

1: For each flow, collect the flow and port statistics.
2: **for** $i = flow_1$ to $flow_n$ **do**
3: Collect all the flow and port statistics from the switches into a file to create the dataset.
4: To collect the flow statistics OFPFlowStatsRequest method is invoked and the required flow statistics are invoked from the switch at a regular interval.
5: To collect the port statistics OFPPortStatsRequest method is invoked and the required port statistics are invoked from the switch at a regular interval.
6: Annotate the Dataset with Attack traffic as 1.
7: Annotate the Dataset with Normal traffic as 0.
8: **end for**
9: **return** Annotated Dataset.

Figure 3.4 shows the process of dataset creation. Here, stepwise phases of dataset creation are shown. After creating and annotating the dataset, machine learning algorithm are trained and tested on the dataset. Before splitting the dataset, various pre-processing techniques have been applied to dataset. Dataset is explored by plotting various univariate and bivariate plots to understand the relationship between the independent and dependent variables. Pre-processing techniques applied include handling missing values, null value removal, categorical value encoding, etc.

Table 3.2: Number of instances in the dataset

Traffic class	Number of instances
Benign	63561
Malicious	40784
TCP Traffic	29436
UDP Traffic	33588
ICMP Traffic	41321

Table 3.3: Traffic category of each traffic instance.

Traffic class	Benign	Malicious
ICMP	24957	16364
TCP	18897	10539
UDP	22772	10816

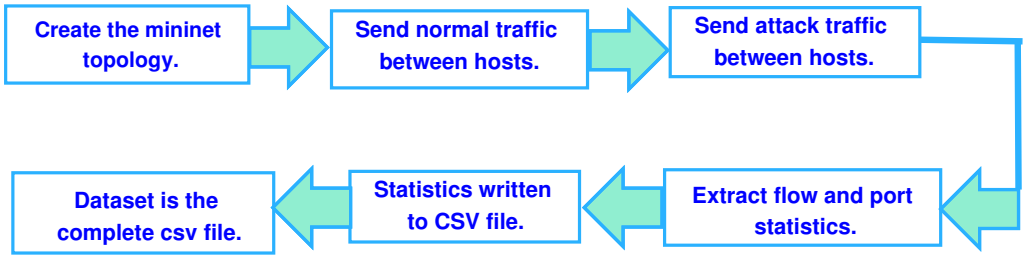

Figure 3.4: Block diagram- Dataset Creation

Feature selection is a key step to choose the important features and get rid of the unimportant ones, which may help the dataset become more precise. We used the correlation matrix as one of the feature selection techniques and eliminated characteristics that did not significantly contribute to the dataset. These attributes, which include Total time, Byte per flow, and Date, do not directly influence the forecast of traffic type. The final variables must be encoded once the feature selection procedure has been applied.

Various categorical features are present in the dataset such as Source_IP, Destination_IP that needs to be encoded. Other features require normalization in which very high valued features and low valued features are scaled to values between (0,1). Before preprocessing, the original dataset comprised 23 columns and 1,04,345 rows. But after preprocessing, the dataset comprised 67 columns due to the addition of dummy variable encoding for categorical variables present in the dataset.

The train and test split are done in the ratio of 80:20 and the distribution of the normal and attack traffic present in the dataset is shown in Table 3.2.

3.3.1 Machine Learning Techniques Used

Machine Learning algorithms can learn from the features in a dataset. It is known as training the machine learning model. The trained model can classify the traffic into benign and malicious

classes. The trained model can also be deployed in real-time to classify the traffic. Some models which perform well in classification tasks as per the literature are mentioned below:

- Logistic Regression [56]: A classification algorithm in Machine Learning which uses a sigmoid function to classify the input into one of the possible classes. It uses a characteristic equation of the form:

$$P(X) = e^{b0+b1*x}/1 + e^{b0+b1*x} \tag{3.7}$$

  Here P(X) is the probability of X which is the output, b0 is bias and b1 is the coefficient associated with the input value (x). The classification label is set to 0 when the probability of an event is less than a given threshold value else it is assigned class label 1. The algorithm is mostly used in the cybersecurity domain for classifying the cases as fraudulent or not. Similarly, the author has tried to predict the likelihood of traffic as Benign or malicious. Besides, the algorithm can also be used for multinomial classification problems. This algorithm produces 83.6% accuracy with the proposed dataset. It is mostly used in the literature, so we also tested the model on our dataset.

- Support Vector Classifier [57]: The algorithm can be used for both regression and classification tasks. But mostly it works well for classification tasks. Various data points are visualized and a decision boundary or plane segregates the class of data points apart. The decision plane can be a straight line or a higher dimensional plane, depending on the number of data points. The points nearest to the decision plane are known as support vectors which help to calculate the margin of the decision plane. The decision plane with large margin is better than a small margin. The optimal decision plane is a generalized decision plane and best segregates the different classes. The decision to apply it to our problem is because it works well for classification problems with a large number of features as in our case. But it does not give satisfactory results with our dataset as the features present in our dataset are highly correlated and finding a decision boundary with such features is not possible. This model, when we train on our dataset yields an accuracy of 85.8%.

- K-Nearest Neighbour [58]: It is an unsupervised learning algorithm that works on the principle that similar things exist together. The algorithm works by predicting the label of the new test data by calculating the distance between the test data and other neighbor

training samples. The distance is calculated by using the equation mentioned below:

$$distance_{a,b} = \sqrt{(x_2 - x_1)^2 + (y_2 - b_1)^2} \tag{3.8}$$

This distance is known as Euclidean distance and (x_1, y_1) & (x_2, y_2) are coordinates of two points in space. Other distance measures can also be used like Manhattan Distance, Minkowski distance. When the distance between different neighbors and test data point is found to be minimum, the particular neighbor is selected and its label is assigned to the test sample. This algorithm, when trained with our dataset attains an accuracy score of 95.22%. This algorithm produces significant results as it uses the statistical parameter of distance measurement to compute the similarity.

- Random Forest [59]: In this method, different decision trees are trained on the dataset.

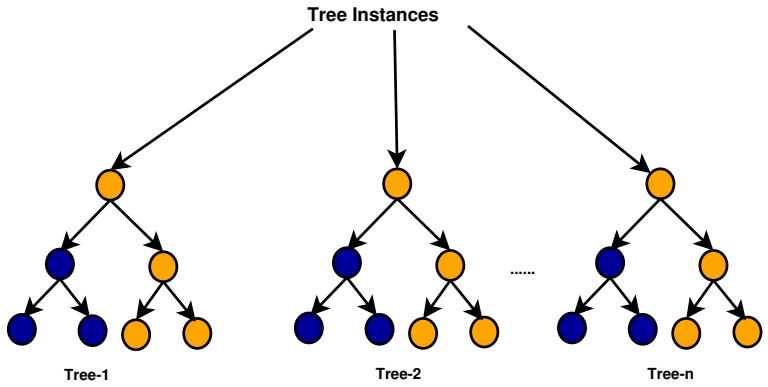

Figure 3.5: Block Diagram depicting Random Forest Algorithm

It outputs a class that is the majority vote of the various decision trees. A large number of decision trees are used for final classification results.

A large number of decision trees participate in decision making, the method is also known as an ensemble of decision trees which is shown in Figure 3.5. It shows that the final output will be based on majority votes of the participating decision trees. It is well suited for a dataset that has a large number of features. This algorithm when trained on our dataset yields an accuracy of 97.2%. Random forest provides accurate predictions as random features are selected during training for each decision tree. Our dataset has a large number of features which increases from 23 to 67 after preprocessing phase and random forest produces significant results with a large number of features. We also apply

k-fold cross validation during splitting to reject overfitting.

- Ensemble Classifier [60]: This classifier considers several models for the task of classification.

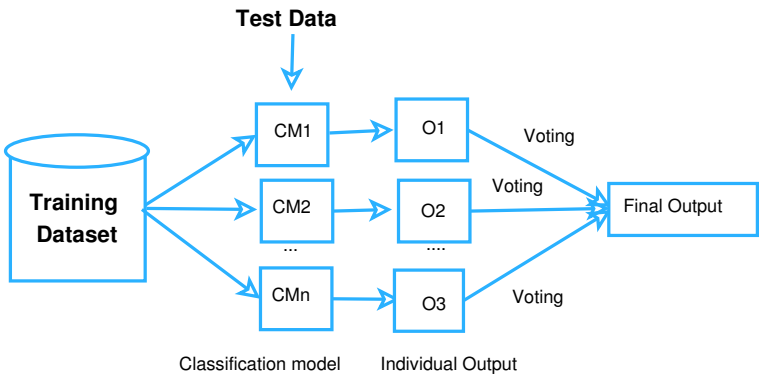

Figure 3.6: Block Diagram depicting Ensemble classifier

In this method, the votes of different classifiers are considered as shown in Figure 3.6. It shows that different classification models (CM_1,CM_2,...CM_n) were considered and trained on the dataset but the final output is decided based on the maximum votes for a class. For eg: if there are three classifiers A, B, and C. A and B predicts the class label as 1 and C predicts class label as 0. The final decision will go with majority votes i.e class 1. The various classifiers include Random Forest, Decision Tree, KNN and SVC. This method attains an accuracy of 97.5% as shown in Figure 3.13. This model produces significant results on our dataset as the classification error is reduced by averaging the error of the different classifiers used.

- Artificial Neural Network(ANN): It is a network of neurons that duplicates how humans think and reason. It consists of many layers a) Input Layer which consists of a set of input neurons. b) Output layer which consists of a set of classes to which input neurons are mapped. c) Hidden layer consists of computations for fine-tuning the weights in the input layer to minimize the error. The inputs from the input layer are passed to the hidden layer. Each connection is assigned weights and each weight gets multiplied with input neurons and bias is added to them as per the equation below:

$$I = \Sigma_{i=1}^{i=n} w_i * x_i + b \tag{3.9}$$

The value of I is passed on to the activation function to select the neuron for feature extraction. The same process follows for other hidden layers and the output layer gives the class probability as output. This method achieved a significant accuracy of 98.2% as shown in Figure 3.13. But this is a black-box approach to attack detection. The random choice of hyperparameters tuned resulted in a significant accuracy. The same hyperparameter chosen may produce a different result on different SDN platform.

- Hybrid Machine learning Model (Support-vector-classifier and Random Forest): In this model, one or more machine learning models are combined to overcome the disadvantage of one another. The performance of individual classifier is compared with the hybrid model which produces better results.

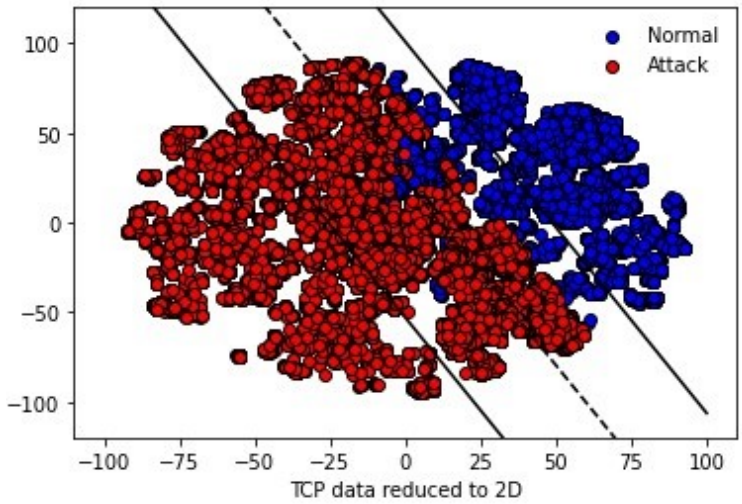

Figure 3.7: TCP data classified with Linear SVC

The classification decision by SVC results in some points being misclassified near the hyperplane. The result of SVC classifier contain both correctly and erroneously predicted results. So, these are further processed by Random Forest classifier which acts as secondary classifier. Figure 3.7 shows the dataset has been reduced to two-dimension by the application of PCA and later by t-distributed Stochastic neighbor Embedding(t-SNE). PCA is initially used to reduce the dimensions to twenty. But it only works for the linear dataset and does not work for a non-linear dataset. So, after applying PCA on the dataset, t-SNE is applied which preserves local distances between the points in high and low dimension and reduced the dimensions to two. After the dimensions are reduced, SVC is fit

to the dataset, which is shown above. Data with unique protocols is fit to separate linear SVCs. But for the points which cannot be inferred to lie in any one of the classes are classified by using the inference from Random forest.

3.4 Experimental Work and Results

The work has been carried out on HP EliteBook with 16 GB RAM and 64-bit processor on windows 10. Figure 3.8 shows the simulation environment where the single Ryu controller is connected to the open-vswitch and the vswitch is connected to the various hosts. Benign traffic is generated from random hosts with the help of mgen tool at the rate of 450 packets per second. The benign and malicious traffic is generated with a specific packet size for a specific duration and are mentioned in Table 3.4.

Table 3.4: Simulation Environment parameters

S.No	Parameters	S.No	Parameters
1	Host OS: Windows10	9	Protocol Used : Open-Flow
2	Guest OS: Ubuntu16.04	10	Graphical package: MiniEdit
3	VirtualBox: 5.1.26	11	Traffic Generation tool: mgen, hping
4	Emulator: Mininet	12	Controller Port Number : 6653
5	Controller: Ryu	13	Simulation Time : 250 minutes
6	Number of Controller: 1	14	Statistics collection interval: 30 seconds
7	Number of Switches: 9	15	Bandwidth plot interval: 30 seconds
8	Number of Hosts: vary depending on topology	16	Number of topologies: 10

The traffic statistics are collected and written in a CSV file. However, the malicious traffic is generated from spoofed IP addresses at the rate of 450 packets per second. The spoofed IP address is used by the attacker at the same packet rate as the benign traffic. The benign and malicious traffic is generated in successive batches by random hosts. The various statistics are also collected from the switches but they show different behavior compared to benign traffic.

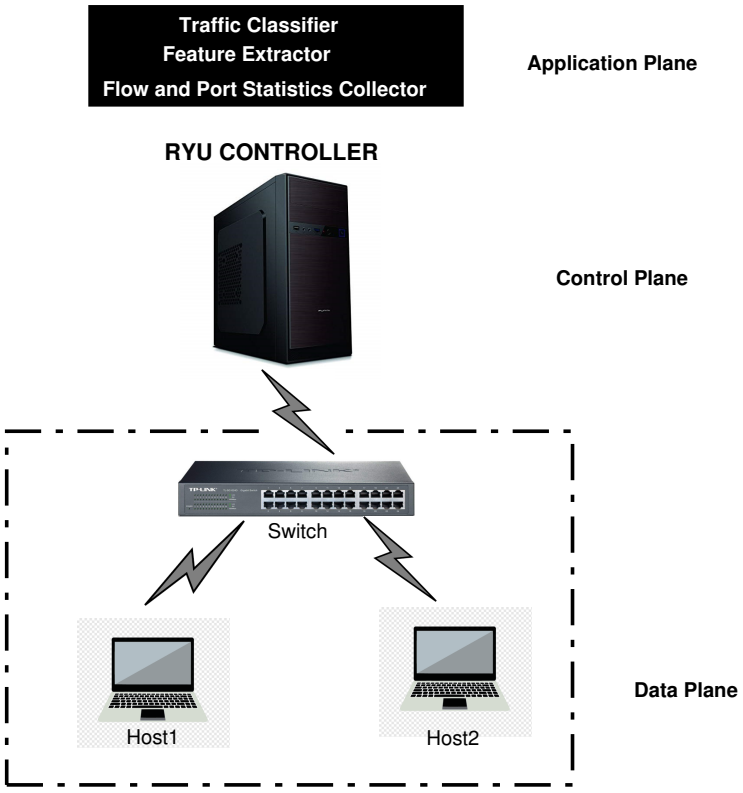

Figure 3.8: Experimental Setup for DDoS attack Detection in SDN

The various flow level statistics are collected by calling the OFPFlowStasRequest method in the Ryu application and port statistics are collected by calling the OFPPortStatsRequest method after every 30 seconds. We choose an experimental threshold for monitoring interval. The chosen value of the monitoring interval is 30 seconds because it is the optimal experimental value at which we get the lowest false positives. But as SDN supports programming, this can be changed as per the network environment while sending the network traffic. Statistics request messages are handled as events and the response is handled by the corresponding reply handler. OFPFlowStatsReply includes statistics such as Packet count, byte count, duration_sec, duration_nsecs, Source IP, and Destination IP. OFPPortStatsReply includes statistics such as Port number, tx_bytes and rx_bytes. The dataset is created with 1,04,345 rows. Benign traffic is generated at the rate of 450 packets per second using mgen tool [61] and attack traffic is generated using hping3 [62] to attack the target host. A log of the features collected from the switches is created in the CSV file which is accessed as a data repository on Mendeley. Figure 3.9 shows the process of training and testing the machine learning model on the dataset. Benign and malicious traffic is generated for 1500 seconds for ten different topologies which is approximately equal to (15000/60) 250 minutes of data and 1,04,345 CSV rows. Each row of the dataset is com-

prised of 23 features. Mininet is used as an emulator on which network topology is emulated and traffic is logged.

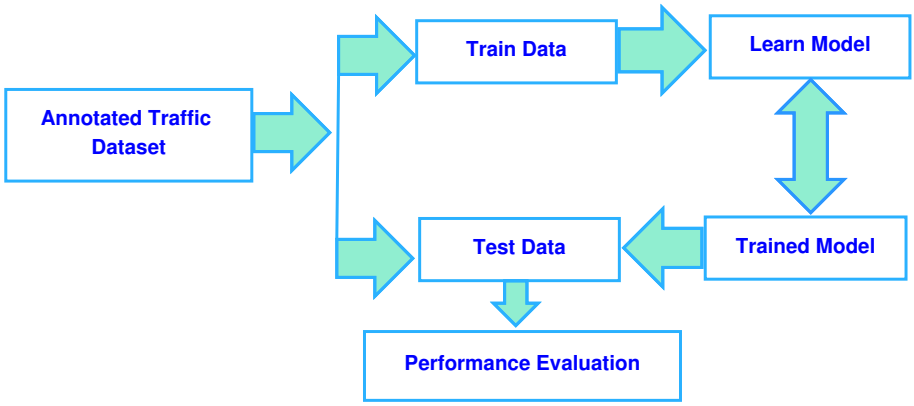

Figure 3.9: Traffic Classification Using Machine Learning

Benign traffic generates a combination of TCP, UDP, and ICMP traffic. Attack traffic generates the combination of TCP-SYN attack, UDP flood attack, and ICMP flood attack.

3.4.1 Performance Parameters

Different Machine-Learning models have been trained and tested on our dataset. We evaluated the performance of each model by calculating Accuracy, Sensitivity, Specificity, Precision, and F1-score. A sample confusion matrix is shown in Figure 3.5: Confusion matrix is often used

Table 3.5: Confusion Matrix

	Positive Predictions	**Negative Predictions**
Actual Positive	True Positive	False Negative
Actual Negative	False Positive	True Negative

to calculate the performance parameters of a machine learning algorithm. In a binary classification problem, it is defined as a 2 x 2 matrix that shows the actual and predicted values of the classifier. The four values in the matrix are rather confusing which are explained as below:

a) True Positive t_p: True positive is the value where the model prediction and actual values in the dataset both are positive. i.e., it is the case where the classifier correctly classifies the traffic as benign and malicious.

b) True negative t_n: True negative is the value where the model prediction and actual values in the dataset both are negative. i.e., it is the case where the traffic is correctly classified as

malicious.

c) False Positive f_p: False positive is the error type where the model prediction is positive and actual values in the dataset is negative. i.e., it is the case where the traffic is incorrectly classified as benign.

d) False Negative f_n: False negative is the error type where the model prediction is negative and actual value in the dataset is positive. i.e., it is the case where the traffic is incorrectly classified as malicious.

Accuracy is one of the measures of performance and it is mathematically defined as a fraction where the numerator specifies the sum of true positive and true negative while the denominator specifies the sum of false positive and false negative along with the terms present in the numerator. It is defined by the equation:

$$Accuracy = ((t_p + t_n)/(t_p + t_n + f_p + f_n)) \tag{3.10}$$

It is the evaluation of the traffic where the classifier correctly predicts both normal and malicious classes.

Recall is defined as the measure of the correct prediction in the dataset. It is also known as Sensitivity. For example, if the recall of a model is 0.11, it means that the model correctly predicts the correct class 11% of the time. It is represented as (R) and defined by the equation below:

$$Recall(R) = (t_p/(t_p + f_n)) \tag{3.11}$$

The recall is equivalent to the detection rate which is defined as the measure of correctly detecting the malicious traffic. Any IDS requires a high detection rate. As seen from the results, the Hybrid model has the best detection rate in our case.

Specificity is defined as the measure of the prediction of the negative class in the dataset. For example, if the specificity of a model is 0.11, it means that the model correctly predicts the negative class 11% of the time. It is defined by the equation below:

$$Specificity = (t_n/(t_n + f_p)) \tag{3.12}$$

A high value of Specificity indicates a lower false positive which leads to more accurate results.

The proposed hybrid model has a high specificity value of 98.18% which is a significant result.

False Alarm rate (FAR) is an important parameter to measure the effectiveness of any system. It is the measure of the inaccurate classification when the model classifies the normal traffic as malicious. An intrusion detection system desires to keep FAR as low as possible.

$$FAR = (f_p/(t_p + f_p)) \tag{3.13}$$

FAR is an important parameter that is kept as low as possible. The hybrid model achieves the lowest FAR of 0.020 which signifies the results.

Precision is defined as the percentage of the model prediction out of the total data values about the positive class. For example: if the precision of a model is 0.5, it means that out of the total predictions the model makes, it is correct 50% of the time. It is represented as P and defined by the equation:

$$Precision(P) = (t_p/(t_p + f_p)) \tag{3.14}$$

Precision predicts the fraction of traffic as benign or malicious, which matches its count present in the dataset.

F1-score is defined as the measure where recall and precision both are used. In the case of an unbalanced dataset, we usually calculate F1-score. It is defined by the equation:

$$F1 - score = ((2 * R * P)/(R + P)) \tag{3.15}$$

From Table 3.6, we can infer that the accuracy achieved by the proposed hybrid model is best. But due to the unequal instances of different classes present in the dataset as depicted in Table 3.2, another performance parameters also need to be considered. Other parameters like Precision, recall, and F1-score are also computed in which a high F1-score implies significant results. Hybrid model has attained the best results.

3.4.2 Result Analysis

Evaluation of the system performance is done using the dataset created in Table 3.6 by calculating Accuracy, Precision, Recall, FAR, and F1-score. Accuracy comparison of SVM-RF hybrid model is done against other specified machine-learning models, shown in Figure 3.13. Data exploration is done to understand the data, the number of instances in each class of traffic, Normal

and malicious traffic distribution in the dataset. Table 3.2 specifies the instance count of different traffic classes present in the dataset. Table 3.3 gives the count of the benign and malicious traffic in each traffic class. This summary data helps to comprehend the dataset. Figure 3.10 shows the distribution of the Source IP address used in the dataset. The blue color represents the benign hosts and red color represents the attacker hosts. It reveals that the IP addresses used for sending benign and malicious traffic are picked from the same pool. It means IP address alone cannot be used as an indicator of benign or malicious traffic and also significantly use the various features presented to detect the attack.

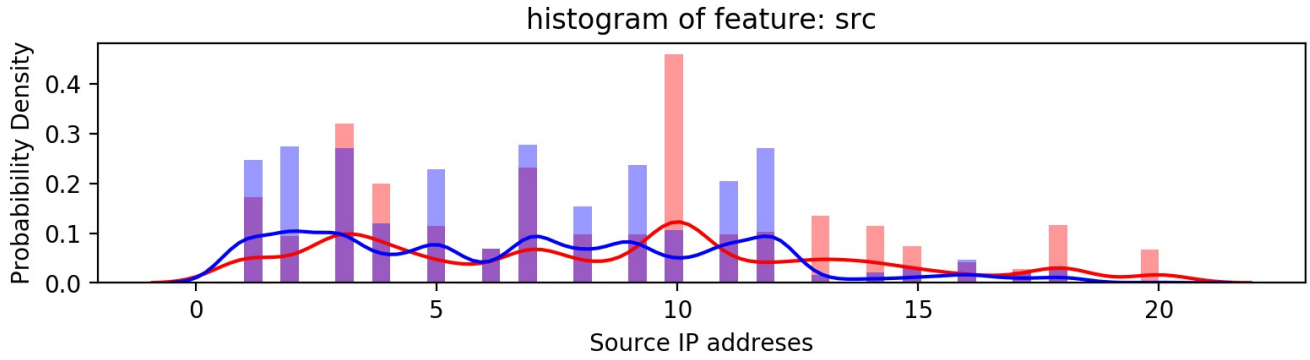

Figure 3.10: Distribution of Source IP address in Normal and Attack Traffic.

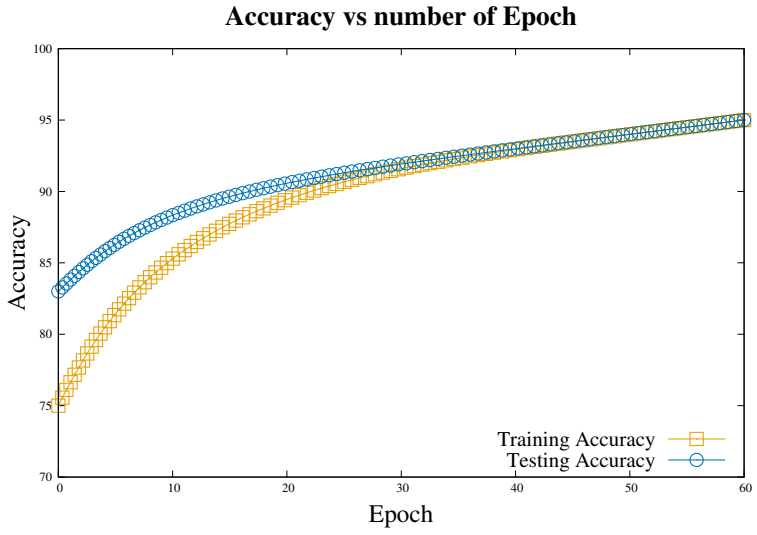

Figure 3.11: Accuracy vs Number of Epoch

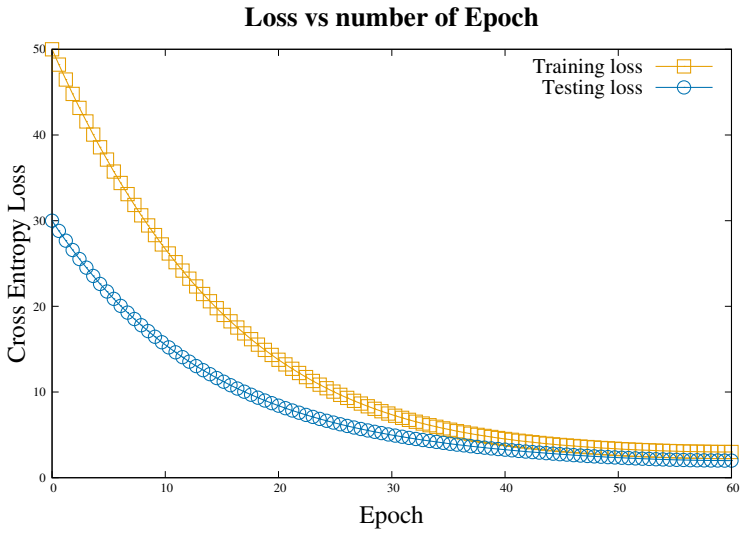

Figure 3.12: Loss vs Number of Epoch

Figure 3.11 shows the Artificial Neural network (ANN) achieved an accuracy of 98.2%. ANN has considerably high accuracy as compared to other machine learning models [63] but it is a black-box approach to attack detection. The random choice of the hyper-parameters which are tuned, helps to achieve this accuracy score. Hyperparameters chosen depend on the used SDN setting. Figure 3.12 shows that cross-entropy loss decreases with an increase in the epoch. It shows that as the number of times the model is trained on the dataset, the error in classifying the traffic decreases.

Table 3.6: Performance Measures of different Algorithms

Model	Accuracy	Detection Rate	FAR	Specificity	Precision	F1-Score
Logistic Regression	83.69%	82.46%	0.175	83.97%	83.31%	82.26%
SVC	85.83%	87.46%	0.125	84.04%	85.79%	86.61%
KNN	95.22%	94.37%	0.056	92.34%	96.83%	95.58%
Random Forest	97.2%	95.45%	0.045	94.56%	96.56%	96.23%
Ensemble Classifier	97.5%	96.43%	0.036	95.32%	96.43%	96.72%
ANN	98.2%	97.84%	0.022	97.43%	97.43%	97.12%
SVC-RF	98.8%	97.91%	0.02%	98.18%	98.27%	97.65%

Figure 3.13 shows the accuracy comparison of different classifiers. It clearly shows that the hybrid SVC-RF model has achieved the highest accuracy in classifying the traffic.

Logistic Regression (LR) model provides an accuracy of 83.69%. It does not offer satisfactory results because the model is not capturing the correct linear relationship between the

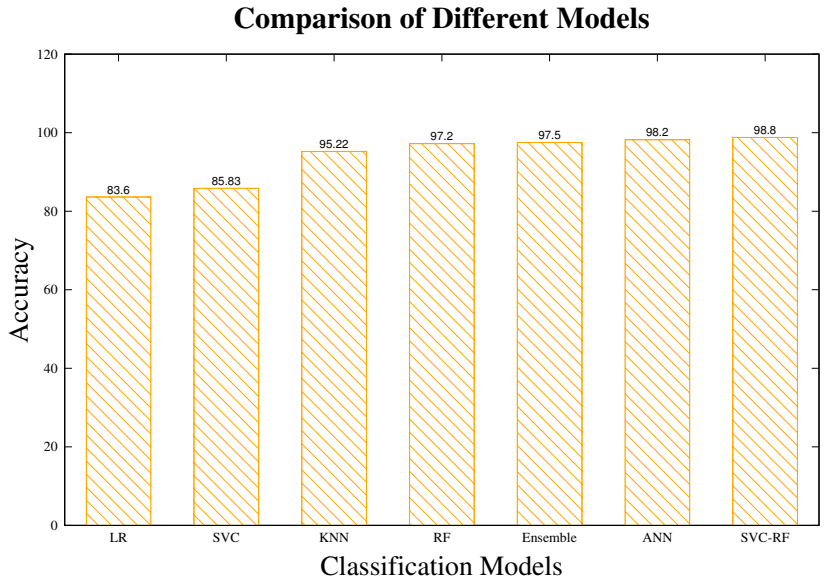

Figure 3.13: Comparative Analysis of ML Algorithms Used

features and so incorrectly predicted the class labels. It is also widely used in the literature.

K-nearest neighbor (KNN) provides a classification accuracy of 95.22%. KNN is a simple classification model which depends on simple calculations. Calculation of euclidean distance between the neighboring point and target point.

Support Vector Machine (SVM) provides a classification accuracy of 85.83%. SVM also does not offer satisfactory results because it finds the decision plane with the help of support vectors having the largest separation from both classes. However, the features present in the dataset are highly correlated and a perfect decision boundary without overfitting might not be always possible.

Random Forest (RF) is a classifier that uses different decision trees for final classification. If one decision tree makes a wrong decision then, other trees can compensate for the wrong decision. Each decision tree provides the classification result and the maximum votes are considered to suggest the final classification result. Hence, Rf proves to be a better classifier.

Ensemble classifier (EC) is a classifier that uses several classifier to make a decision. The different classifier which are used includes KNN, Decision Tree, Random Forest, and SVC. The accuracy achieved by the classifier is 97.5%, which is substantially better than individual clas-

sifier performance.

Support-Vector-classifier with Random-Forest (SVC-RF) hybrid classifier: This classifier is used as a combination of two machine learning algorithms and gives the best results with our dataset.

3.4.3 Comparative Analysis with Existing Results

To evaluate the proposed method, Table 3.7 shows a comparison with the existing work done in the area of DDoS attack detection using emulated dataset. The top existing benchmark result is found to be 96%. From the Table, it can be seen that our proposed model achieves the highest accuracy of 98.8%. The hybrid approach followed for the proposed model plays a significant role in attack detection.

Table 3.7: Comparison results of traffic classification using various simulated SDN Datasets

S.No	*Authors*	*Testing Accuracy*
1	Meti et al. [15]	80%
2	da Silva et al. [2]	88.7%
3	perez et al. [64]	95%
4	Ye et al. [65]	95.24%
5	Ko et al. [66]	96%
6	han et al. [67]	96%
7	myint et al. [48]	97%
8	**Proposed**	**98.8%**

Dataset for unique protocols is fit to linear Support-Vector-Classifier (SVC) [68] and for the suspicious points, inference from the Random forest is used for final classification. This model proved to be the best performing model for our dataset with reduced training time as well.

3.4.4 Observation and Discussion

The results shown above clearly indicate that a network administrator of an organization can easily use in his capacity the machine learning model developed for early detection of the attack. The DDoS attack can be detected by implementing the proposed machine learning model. The experimental work is done on the SDN dataset created on the emulator, the software version of the real switch, so the results will be valid in real-time. The features proposed in the dataset

can be collected using the proposed method in real-time and the machine learning algorithms can be deployed for classification. The hybrid model of Support Vector machine and Random Forest has produced high precision and recall value which is a promising solution in real-time. SVC-RF model works well as the misclassification of points done by SVC is taken care of by RF.

Various machine learning algorithms used show promising results for detecting the attack. The proposed hybrid model shows highest results in terms of various performance parameters used because of the novel and significant features which we have proposed in our dataset. Also, the proposed machine learning technique of SVC filtered by Random forest help find the decision boundary which best fits the data and achieve significant results. Previous work which has been done does not achieve significant results as shown in Figure 3.13, which yet proves the significance of our results. Previously, attack detection is done by using some statistical approaches, which we also implemented in our previous work [69] which involve analysis of the various flow and port statistics. But now with the created SDN dataset, we can use promising technologies of Machine Learning and Deep Learning to classify the traffic.

3.5 Summary

An innovative DDoS countermeasure is covered in this chapter. DDoS attacks are now causing havoc in organisations. Despite several countermeasures, the attack has not yet been prevented. In this chapter, we tried to use machine learning to address the issue of DDoS attack detection. The hybrid machine learning model of Support vector classifier with random forest produces the highest classification accuracy of 98.8%.

ARP-based attacks are vulnerabilities exploited by the attacker at the Data-Link layer, which can lead to MITM and Eavesdropping attacks. Arp vulnerabilities that exist in traditional networks exist in SDN as well. Existing solutions to ARP-based attacks in SDN are either statistical or complex cryptography techniques that are both infeasible and computationally intensive. As a result, in the following chapter, the author proposed a novel method for detecting ARP-based attacks.

CHAPTER 4

ASCERTAIN THE EFFICIENT MACHINE LEARNING BASED APPROACH TO DETECT DIFFERENT ARP ATTACKS

The previous chapter discussed the detection technique for DDoS attacks, which is primarily based on the use of a machine learning algorithm. However, there are other vulnerabilities in traditional networks that occur in SDN. This chapter discusses the ARP vulnerabilities that exist at the Data-Link layer of a network. SDN provides a novel programming paradigm for detecting attacks, which motivates the development of a solution for dealing with ARP vulnerabilities in the SDN environment. This chapter proposed a novel method for detecting ARP-based attacks [70]. The authors also generated traffic datasets for benign and attack traffic (ARP Poison, ARP Flood) to achieve the objective of attack detection. This dataset was used to detect ARP poison and flood attacks using machine learning techniques.

4.1 Introduction

Although SDN provides several advantages due to the logically centralized controller, it is still highly susceptible to traditional attacks such as Address Resolution Protocol (ARP) Poisoning, ARP Flooding, and others. ARP is an address resolution protocol that provides the Media access control (MAC) address of a host, from its IP address. Each host in the network holds an ARP table that maps IP addresses to MAC addresses. In a traditional network, when the source host (H1) communicates with the destination host (H2), the source host checks its ARP table for the destination MAC address. If IP/MAC pairing is not present in H1, it will broadcast an ARP request packet, which will be answered by the destination host (H2) as shown in Figure 4.1.

The source host (H1) revises its ARP table with the destination MAC address of the Host (H2) and the communication proceeds. But if the attacker interferes and maliciously updates the ARP table then it can lead to an attack situation known as an ARP Poison attack [71].

ARP Poison attack: It is an attack where the ARP table reflects malicious information, propagated by the attacker by sending a fabricated ARP request or reply packets using Scapy [72]. ARP

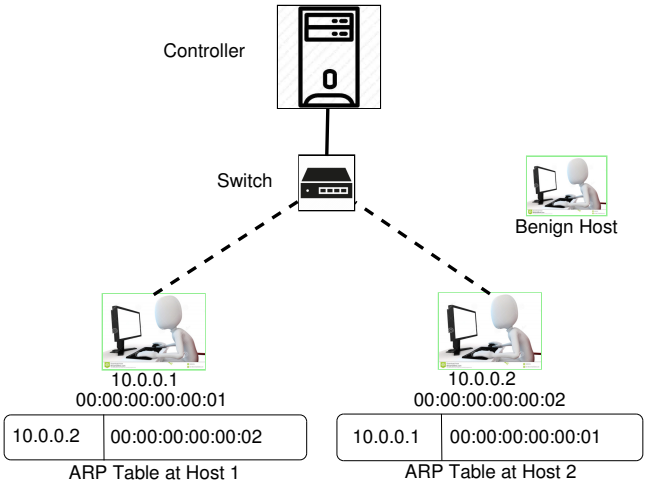

Figure 4.1: Discovering a host during ARP Process

Poison attack can be performed in many ways and two are discussed below:

1) Crafting an ARP reply to a genuine host's ARP request: In this method, the attacker keeps on waiting for the ARP request packet from the genuine host for which it will craft the ARP reply packet. While crafting the ARP reply, the attacker put its MAC address in the destination MAC address field. So, every packet which has to reach the genuine host will now reach the attacker.

2) Crafting an ARP request to a genuine host: In this method, the attacker broadcasts a forged ARP request packet, to which legitimate hosts respond, resulting in false MAC/IP information being stored in the ARP table and the table being poisoned. ARP Poisoning and ARP Flood are the means to carry out the Eavesdropping and MITM attack [73].

ARP Flooding attack: It is an attack that uses random host machine MAC addresses to flood the host's ARP table. Random hosts flood the ARP table to capacity, causing delays in the processing of legitimate requests.

These attacks can be mitigated by establishing a secure link [10, 74]. The proposed solution for preventing the attack is to block the specific port identified programmatically, which is efficient because it avoids handcrafted feature construction and is thus efficient in terms of time and processor load in mitigating the attack.

Existing solutions are based on either checking the traffic against the stored MAC/IP binding which becomes a time-consuming task when the network is a significant one, checking the pattern of traffic which is time taking task [20], cryptographic solutions are complex task in terms of processing power, creating a flow-graph for detection of ARP-Poison attack [75] or statistical techniques which is also a computationally intensive task. This motivates the author to provide

a novel solution to ARP-based attacks in the SDN environment. The research contribution of the work are briefly outlined below:

- SDN traffic dataset for ARP-Poisoning and Flood attack is created with the help of a mininet emulator [76]. There are no publicly available datasets for this attack. Previous work has created the experimental data but has not made it public. The proposed dataset and code for traffic classification are available in Mendeley Data repository[1].

- Application of different Machine Learning algorithms is done to classify the network traffic into one of the three classes (Benign, ARP Poison attack, ARP Flooding attack). The trained ML algorithm on the proposed dataset is implemented as an application at the controller to categorize the traffic.

- Attack detection on hosts is carried out by analyzing the significant features present in the dataset with the machine learning model. The machine learning model helps to detect the attack in less time as shown in Figure 4.8 compared to other statistical methods [77].

4.1.1 Previous work vs Proposed Method

The work on ARP Poison and ARP Flooding has been done in the past also. However, the approach and techniques followed previously differ significantly from the proposed technique:

Sebbar et al. [78] proposed the detection of MITM by analyzing the delay during which the node responds to the controller. If the node responds within a threshold time, it is authenticated, otherwise, it is a delay attack. The above approach used a threshold-based approach with analysis of only one parameter but the proposed approach used seventeen significant parameters which provide promising results. Cheng et al. [9] proposed Round-Trip-Time's (RTT) minimum, average, and maximum cases for ICMP traffic classification. Four different wired and wireless mediums were considered for the controller, attacker, and genuine host. The accuracy in all four cases came out to be 70%, 98%, 87%, and 51% respectively. The proposed approach in our work attained better accuracy of 99.73%. Dhawan et al. [75] proposed the various flow statistics including packet_in count, the number of packets sent, switch-id, port-number, and IP/MAC binding information via flow graphs. These flow graphs are monitored continuously against the already defined policies and those learned over time to detect the attack. The approach proposed

[1] `https://data.mendeley.com/datasets/yxzh9fbvbj/1`

in our work analyses mostly similar statistics with more promising results when using the ML approach.

Hsiao et al. [79] proposed a ML approach for ARP attack detection. The data was collected from the National University of Kaohsiung by simulating a network of 256 hosts. The dataset was not published for verifying the results whereas the proposed approach made the data-set public. Jaspreet Kaur [77] proposed three different methods (including signature-based method, manual Wireshark packet analysis method, and ML method) for ARP spoofing attack detection. Naive Bayes algorithm achieved the lowest FAR and highest accuracy of 93%. But the proposed method achieved a better accuracy of 99.73%. Ma et al. [80] proposed the Bayes method for determining the probability of an attack. Based on the probability, it used different ML algorithms for attack detection. With only four features and no experimental data proof, the author detected the attack. Our work achieved quite a good accuracy with dataset proof for public validation.

4.2 Materials and Methods

In this section, the significant features generated during the dataset creation along with the proposed methodology are discussed. The dataset generated during the first phase is later used for network traffic classification. The ML algorithm uses the features in the dataset to classify traffic, and normal traffic is separated from attack traffic.

4.2.1 Dataset Creation

An efficient dataset contains the adequate features required for early attack detection. A topology is emulated in mininet and Internet Control message protocol (ICMP) traffic is sent through the different benign hosts as shown in Figure 4.2. Some nodes are assumed as malicious nodes which send crafted ARP requests to poison the ARP table of genuine hosts. During the experimental setup, firstly the topology is created in Mininet.

Secondly, the ARP table of the host is checked and the ARP table of each host contains the last ping host MAC Address. Then a Scapy script is written to perform the poisoning of the ARP table. After running the attack simulation, the genuine host's ARP table is updated with malicious MAC-ID credentials.

It demonstrates that the IP address (genuine host) is linked to the attacker's MAC address. In this manner, an ARP Poisoning attack is carried out, in which if host A sends traffic to host B, it is routed to host C. The proposed dataset is created by writing a python application that runs

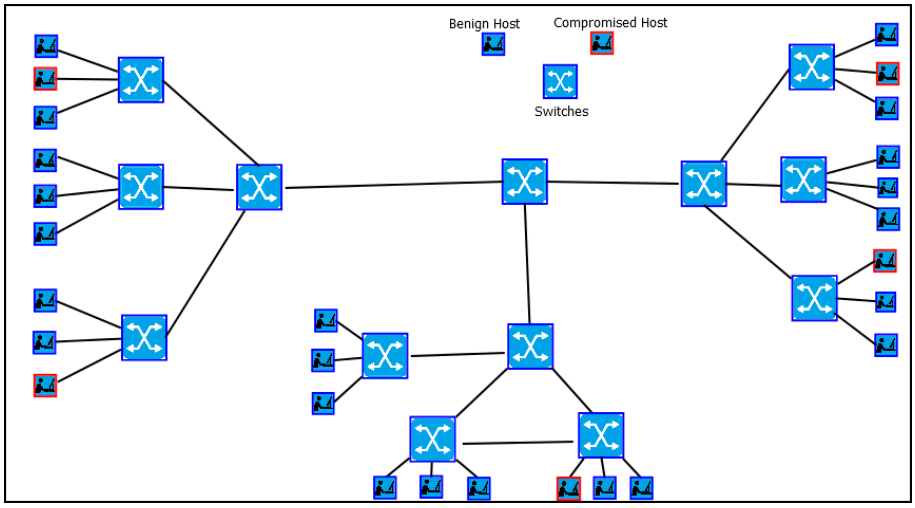

Figure 4.2: Experimental Tree Topology with depth and fanout value of 3

over the Ryu controller and uses the Ryu Application Programming Interface (API).

Table 4.1: Features used in the Dataset

S.No.	*Features Used*	S.No	*Features Used*
1	Source MAC Address at ethernet	9	Switch-ID.
2	Source MAC address at ARP header	10	Source port
3	Sender IP Address in ARP request.	11	Ping statistics
4	Target IP Address in ARP request.	12	Destination port
5	Protocol Code	13	in_port, out_port
6	Destination MAC Address at ARP	14	Operation Code
7	Destination MAC Address at Ethernet	15	Round trip time.
8	Time to live (TTL)	16	Packet loss
17	Number of Packet_in messages		

By invoking the packet_in handler method and matching the ethernet-type field, the application collects the features present in Table 4.1. If the ethernet-type field matches the ARP packet, we extract the ARP header information and write each field, including switch-id, in port, out port, source MAC address in ethernet, destination MAC address in ARP, source IP, destination IP, and operation code. If the ethernet type matches an IP packet, we determine whether it is ICMP, TCP, or UDP traffic and extract the matching header fields, which include the above fields in addition to Source Port, Destination Port, and Time-To-Live (TTL). Another python application extracts the Round-Trip-Time (RTT) and writes them to the CSV file. These CSV files are combined to form the dataset, which is based on the common Date-Time field.

The dataset is a CSV file with $1,34,000$ rows and seventeen features per row. The data is collected for a total of 30 minutes. The count of data instances in the dataset is summarised in

Algorithm 4 Proposed Algorithm for ARP-based attack detection and traffic classification.

Input: Traffic statistics of topology attack.
Output: ARP attack classified traffic.

Initialization: Source MAC address, Destination MAC address, Sender IP address, Destination IP address, Switch ID, in_port, out_port, Protocol code, Number of Packet_in messages, Operation Code, packet loss.
For each flow extract the features as mentioned above.
Python application executes at the controller and different features are extracted.
for $i = Packet_1$ to $Packet_n$ **do**
ARP header→ARP features
IP header→ IP features
Benign traffic features → CSV
ARP Poison Traffic features → CSV
ARP flood traffic features → CSV
Assign the Benign traffic label as 0.
Assign the attack traffic label as 1.
end for
Apply Machine Learning Algorithm to the labeled training dataset.
(I) Text columns→vectors.
(II) Data Columns→ Normalize.
(III) Dataset→Training and Test dataset.
(IV) Machine Learning Algorithm→Training dataset.
Machine Learning Algorithm→ Classification Results.
return Test Dataset classified into normal and malicious classes.

Table 4.2, with traffic divided into ARP request, reply, and ICMP request, reply packets. The count of ARP request packets and reply packets does not differ significantly in genuine traffic, but it differs significantly in ARP flood and Poison attacks due to attacker meddling.

Table 4.2: Message-wise categorization of Dataset.

Traffic Class	***Benign***	**ARP Poison**	***ARP Flood***
ARP request	16749	4622	92138
ARP reply	16600	3397	0
ICMP request	178	129	0
ICMP reply	172	15	0

4.2.2 Proposed Methodology of Traffic classification using Machine Learning Algorithm

The proposed algorithm is shown in Algorithm 4 and illustrates the steps followed for dataset creation and further classifying the traffic by using machine learning algorithms. Our approach is similar to that of [20], but the proposed approach has logged the various features and created a dataset. The proposed work is distinguished from this work by the use of machine learning

algorithms on the created dataset to detect the attack.

Different classes of traffic include benign Traffic, ARP Poison attack traffic, and ARP Flooding attack traffic. Different ML models are discussed and justified for their performance on the proposed dataset.

- Logistic Regression (LR) [81]: This algorithm does not perform well for the proposed approach as it considers the linear relationship between the variables which does not work in the proposed data where some data points are also inversely related. This algorithm, when trained with our dataset produces 56.2% accuracy.

- Naive Bayes Classifier (NBC) [82]: The classifier is based on the Naive Bayes theorem which assumes independence among the features to identify the class label. The assumption of independence among the various features in Naive Bayes resulted in low performance on the proposed dataset. The dataset features are highly correlated to each other, so this classifier does not perform well. For example, the malicious traffic has the same packet rate as normal traffic but it shows the high value of Packet_in messages. This algorithm provides an accuracy score of 64%.

- Support Vector Classifier (SVC) [83]: The classification algorithm provides an accuracy of 85.7%.

- Artificial Neural Network (ANN) [81]: ANN is made up of many layers with the help of which it learns the pattern present in the data. It provides an accuracy of 93.65%.

- Decision Tree (DT) [84]: The algorithm starts from the root of the tree and checks the value with the other node. The decision is taken either to move left or right based on the comparison result. It checks the value of each feature before taking the decision. It captures the dependency between the various features present in the dataset and achieves an accuracy of 96.37%.

- Random Forest (RF) [84]: In this algorithm, many decision trees perform the classification task. In RF, the accuracy is averaged out for a good performing DT and poor DT and thus performs best. It is well suited for the proposed dataset with the significant features present in it and yields an accuracy of 97.4%.

- Ensemble Classifier (EC) [84]: This classifier takes into account multiple models (DT, RF, NB, SVC). Separate classifiers were trained on the dataset, but the result is determined by the class with the most votes and yields an accuracy of 98.2%.

- Convolution Neural Network (CNN) [85]: CNN is a deep learning algorithm which generally works for image classification. But Conv-1D (One dimensional Convolution Network) produces significant results for use-cases like anomaly detection and recommender system. CNN model refers to a form of neural network with the existence of convolution layers. The convolution layers contain a kernel to extract the features present in the input and the kernel convolves with the input vector to extract the significant features. The convolution operation is mathematically expressed by the equation shown below:

$$O_t = tanh(y_t * h_t + s_t) \tag{4.1}$$

  Here O_t is the output after convolution, y_t is the input, h_t is the weights associated with the kernel, and s_t is the bias associated with the kernel. The extracted features are of high dimension so the max-pooling layer accepts the output of the convolution layer and generates a lower feature dimension as output. When it is applied to the proposed dataset, it produces significant results as it automatically detects the significant features required for classification. Also, Conv-1D does not require much pre-processing. This algorithm yields an accuracy of 98.4%.

- Long Short Term Memory (LSTM) [86]: Recurrent Neural Network (RNN) suffers from an issue called short-term memory. This issue occurs due to the smaller value of gradients used for weight updation. In initial layers, the gradient has the smallest values, due to which the layer does not learn. Thus, during long sequence prediction, RNN has a short-term memory. LSTM evolved as a solution to this issue. It operates through the use of gates (input, forget, and output gate) which are a kind of neural network to assure that relevant information for prediction is not lost. LSTM produces significant results with the proposed dataset as it has time-dependent features. The characteristic equation of the LSTM network is mathematically expressed as under:

$$c(t) = f_t * c_{t-1} + i_t * c_t \tag{4.2}$$

Here, c_t is the state of a memory cell at time t, i_t is the input gate, c_{t-1} is the prior cell condition and f_t is the forget gate result. When applied to our dataset LSTM does not produce significant results as it might not able to extract the relevant features to be used for classification. This algorithm yields an accuracy of 94%.

Table 4.3: Hyper-Parameters of CNN-LSTM Model

S.No.	***Parameters***	**Value**
1	Filters at Conv Layer	64
2	Kernel size	3
3	Activation function at Conv Layer	relu
4	Padding at Conv Layer	1
5	Activation function at Pooling layer	relu
6	Hidden nodes in LSTM layer	128
7	Activation function at LSTM layer	tanh
8	Batch Size	64
9	Learning Rate	0.001
10	Optimization function	adam
11	No. of Epoch	50

- Hybrid CNN-LSTM: The hybrid model has yielded significant results in the past. LSTM can remember the past longest distance input sequence while working on the current input. CNN has the advantage of extracting relevant features automatically. Thus, the combination of appending CNN layers with LSTM yields a hybrid model and provides promising results. Figure 4.3 depicts the architecture of the CNN-LSTM model that was used and outlines the number of layers and kernels used at each step. This algorithm yields an accuracy of 99.73%. Table 4.3 shows the experimental parameters associated with the model. The model requires more epochs to converge because the dataset is bigger with one lakh thirty four thousand rows. It requires 50 epochs of training, after which only the model converges.

4.3 Implementation Details and Results

The implementation environment utilized for performing the experiments is shown in Table 4.4. It shows the various parameters related to the simulation environment which mainly includes the mininet emulator for creating the topology and running the traffic tests using the Ryu controller.

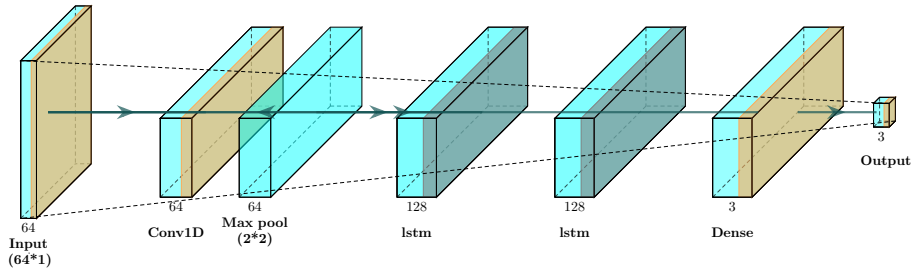

Figure 4.3: CNN-LSTM Model

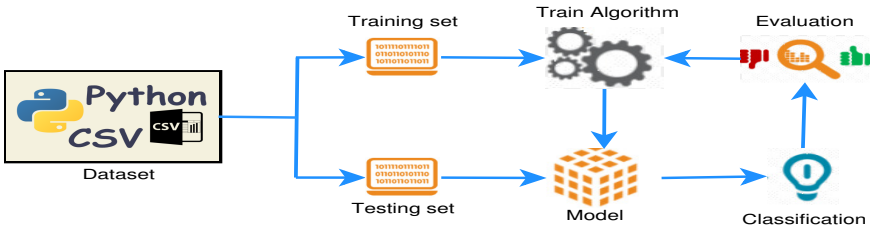

Figure 4.4: Traffic classification using Machine learning

Table 4.4: Simulation Environment Parameters

Parameter	***Value***	**Parameter**	***Value***
Host OS	Windows10	Guest OS	Ubuntu 18.04.
Virtual Box	6.1	Emulator	Mininet.
Controller	Ryu.	No. of Controllers	1.
No. of Switches	Topology Dependent.	No. of Hosts	Topology Dependent
Protocol Used	Openflow	Graphical Package	MiniEdit.
Traffic Tool	mgen, tcpdump	Port Used	6653.
Simulation Time	300 sec	No. of Topologies	Seven.

The number of switches and hosts may vary depending on the selected topology. The benign traffic is generated using the mgen tool and attack traffic is generated via hping3 tool. Benign and malicious network data is generated at the rate of 10 packets per second (pps). Traffic is generated for the duration of 300 seconds for seven different topologies which are equal to 3000 seconds of data for each topology and 21000 seconds of total data equals 30 minutes of data. This data is distributed amongst 134000 rows in a CSV file. The dataset is used for traffic classification in SDN.

4.3.1 Performance Parameters

Different machine learning algorithms have been evaluated by calculating different evaluation measures such as Accuracy, Precision, Recall, and F1-score. In a classification problem, the values from the confusion matrix are used to calculate the prediction matrices.

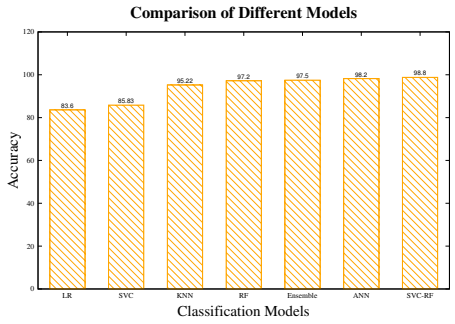

Figure 4.5: Comparative analysis of different Algorithms Used

Specificity is defined as the model prediction of the negative class in the dataset, i.e., if an ML model predicts specificity as 0.81, it implies that ML model prediction about the negative class present in the data is 81%. It is mathematically expressed as follows:

$$Specificity = (tn/(tn + fp)) \tag{4.3}$$

An increase in the Specificity value is a good indicator of lower false positives. CNN-LSTM model attains a specificity value of 98.18% which is an indicator of fewer false positives and hence desirable.

False Alarm rate (FAR) is also one of the significant performance measures. It determines the rate of mis-classification i.e., if the model predicts the benign traffic as intrusive. Any network traffic monitoring system aims to keep the FAR value the lowest.

$$FAR = (fp/(tp + fp)) \tag{4.4}$$

CNN-LSTM algorithm attains the FAR value of 0.018 which significantly evaluates the model performance.

4.3.2 Result Analysis

This section summarizes the results which are attained by applying the CNN-LSTM model to the dataset. Figure 4.5 expresses the accuracy achieved by different ML algorithms. CNN-LSTM model outperforms all other models as the hybrid of CNN and LSTM works well on the proposed dataset.

Figure 4.6 shows the accuracy increase with the number of epoch and attained a significant accuracy of 99.73%. Figure 4.7 shows the loss decreases with the number of epochs. The

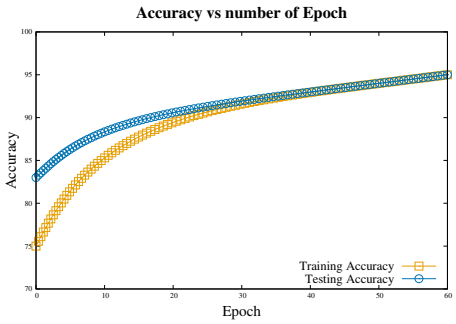

Figure 4.6: Accuracy vs Epoch of CNN-LSTM model

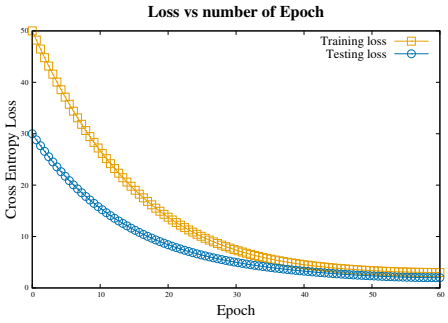

Figure 4.7: Loss vs Epoch of CNN-LSTM model

model is trained each time the epoch increases and can better classify the traffic which leads to a decrease in loss.

Figure 4.8 depicts the time taken to depict the attack. It shows that the ARP Poison attack is depicted within microseconds but the ARP Flooding attack is depicted in seconds of its occurrence.

The impact analysis of the attack is evaluated by analyzing the CPU utilization and memory utilization when the attack takes place.

Figure 4.9 depicts the memory usage by host H1. When the ARP Poison attack takes place, the traffic flow between the benign hosts is diverted to the middle man H1. It rises above the normal usage limit of 2*10-6 MB.

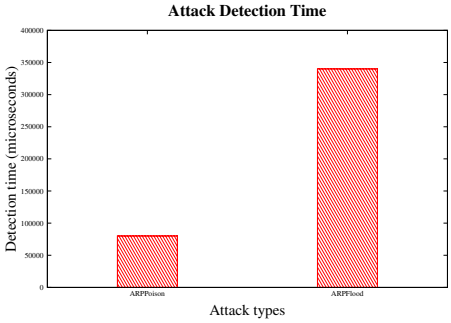

Figure 4.8: Attack Detection time

Table 4.5: Performance measures of different Algorithms

Model	Accuracy	Precision	ARP Poison			ARP Flooding		
			Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
LR	56.2	55.4	54.3	53.4	52.6	55.7	54.3	55.3
NB	64.0	63.4	64.5	63.2	62.3	61.4	63.9	63.6
SVC	85.7	83.8	84.7	86.4	84.5	86.6	85.8	84.9
ANN	93.6	92.8	90.5	91.2	93.2	91.4	92.5	93.4
DT	96.3	94.6	91.2	92.3	94.4	93.1	92.7	95.4
RF	97.2	96.8	97.3	95.6	94.2	93.4	95.4	96.8
EC	98.2	97.7	96.4	97.1	95.7	98.1	97.3	97.9
CNN	98.4	97.8	98.3	95.6	91.2	93.4	95.4	97.7
LSTM	94	92.7	91.4	93.1	91.9	90.5	92.3	93.9
CNN-LSTM	99.73	97.8	98.3	95.6	96.2	97.4	95.4	98.9

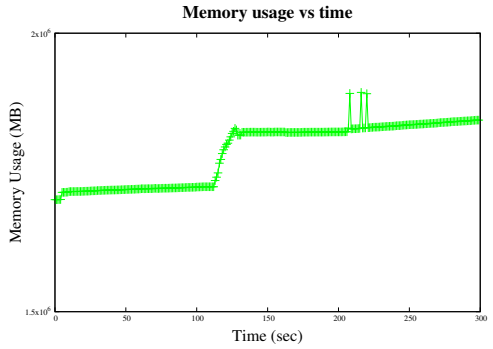

Figure 4.9: Memory Usage Graph

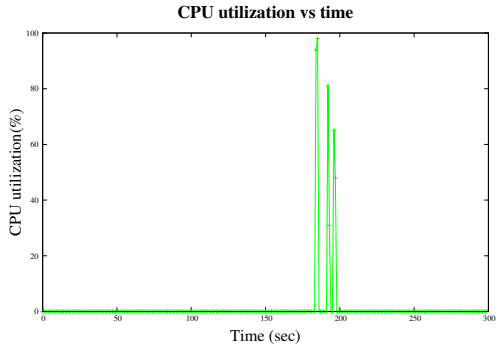

Figure 4.10: CPU Utilization Graph

Figure 4.10 depicts the CPU utilization at host H1. When an ARP Poison attack occurs, the CPU utilization increases to 97%. This utilization increases, due to the attacker which eavesdrops on the traffic between two genuine hosts. Due to this a lot of ARP request messages got accumulated and CPU utilization increased. But when the attack is depicted by the detection module, the corresponding ports are programmed to drop the messages. Due to this, the port utilization of the target host decreases.

Table 4.5 shows the Accuracy, Precision, Recall, and F1-score achieved for attack detection. It can be seen that due to the large gap in the number of instances present in the dataset, other performance measures like F1-score also have to be evaluated and a high F1-score means promising results.

4.3.3 Comparative Analysis with Existing Results

To evaluate the proposed method, a comparison is done with the existing work done in the area of ARP-based attack detection using emulated dataset. The top existing benchmark result is found to be 96%. From the Table, it can be seen that our proposed model achieves the highest accuracy of 99.73%. The hybrid approach followed for the proposed model plays a significant role in attack detection.

Table 4.6: Comparison of proposed work with existing research for ARP-based Attacks.

S.No	Authors	*Testing Accuracy*
1	Hsiao et al. [79]	95.9%
2	Sebbar et al. [78]	98%
3	Jaswinder kaur [77]	93%
4	abdullah et al. [10]	91.2%
5	ma huan et al. [80]	80%
6	cheng et al. [9]	96%
7	**Proposed**	**99.73%**

Table 4.6 compares the proposed work to previous work done by other researchers. The existing highest result shows an accuracy score of 97% and the proposed work shows an accuracy score of 99.73%. The hybrid of CNN and LSTM achieves significant results with the chosen hyper-parameters.

4.3.4 Observation and Discussion

The significant results achieved indicate a promising solution for detecting the ARP attack. A trained ML model can predict the attack class with high accuracy. As the dataset is created using an emulator, the solution will work in real-time with approximately similar accuracy. Various other solutions which already exist either use a non-SDN dataset or have used an experimental dataset that is not accessible to the research community. There are various other approaches including the statistical approach, cryptographic approach, Flow-graph based approach which have been applied for ARP Poison and ARP Flooding attacks detection but they do not provide significant performance. As compared to previous approaches, the ML approach has achieved significant results as can be evident from the high precision and recall value. A topology Poison and Flooding attack dataset is created using mininet. The created dataset is significant because to the best of our knowledge, ARP Poison and Flooding attacks dataset is not available for public use. The trained hybrid CNN-LSTM model performs best, which implies that it has identified key features for attack detection and hence can be deployed in real-time in an SDN environment. But due to the lack of an appropriate SDN dataset, the attack detection has not been done with this method, which otherwise, is the best choice as it produces significant results. In our previous work, the work on the classification of DDoS attack detection using machine learning has been done [39].

Different machine learning models proposed have also shown significant results for attack detection.

4.4 Summary

This chapter discusses a cutting-edge ARP-Poisoning and flood attack defence strategy. ARP-based attacks can now occur in SDN in addition to conventional networks. The atttcak cannot be avoided despite the many prevention techniques. In this chapter, we tried to use machine learning and deep learning to address the issue of ARP Poison and flood attack detection. In an attempt to apply Machine learning and deep learning techniques, ARP Dataset for SDN has been generated. The best performing algorithm of CNN-LSTM classifies the traffic with an accuracy of 99.73%.

The next chapter chooses to follow the use of Deep Learning techniques for DDoS attack detection. The previous work of detecting DDoS attacks using machine learning for feature ex-

traction is time-consuming. It can be made more efficient by using a deep learning technique that automatically extracts the important features for attack detection, as discussed in the following chapter.

CHAPTER 5
DDOS ATTACK DETECTION USING DEEP LEARNING

In Chapter 3, DDOS attack detection is performed using ML algorithms. But the results can be improved using the efficient Deep Learning algorithms. A considerable time was spent on manual feature extraction using machine learning algorithms. While Deep Learning techniques extract the significant features automatically making the process more efficient. Deep Learning Techniques can also improve classification accuracy. This chapter applies Deep Learning techniques using SDN-DDOS dataset for DDoS attack detection. Deep learning algorithms offer significant solutions in a variety of fields, so we tried this technique to improve the accuracy of the results obtained. Existing work [50] for DDOS attack detection using Deep Learning provides significant results for attack detection in the network in which Stacked Auto-Encoder and Multi-Layer Perceptron(SAE-MLP) provide the highest rate of attack detection. This chapter reviews the publicly available datasets where DDOS attack detection is done by implementing different deep learning algorithms using the proposed SDN-DDOS dataset and other publicly available datasets.

5.1 Introduction

SDN is vulnerable to a variety of attacks. Network Manipulation attack, Traffic Diversion attack, Side-channel attack, App manipulation attack, Denial of Service (DoS) attack, Distributed Denial of Service (DDoS) attack, ARP Spoofing attack, API exploitation attack, Traffic sniffing attack, and Password guessing attack are a few examples. Previous work which has been done in this area mainly focused on applying statistical techniques, Machine learning algorithms for anomaly detection in SDN which does not yield significant results. So, the author proposed deep learning algorithms for traffic classification in SDN setup. We aim to classify normal traffic and Distributed-Denial-of-Service (DDoS) [87] traffic. DDoS leads to disruption of the target host services by flooding it with requests from many different sources.

In this work, we have been provided with a customized Software Defined Network (SDN) dataset generated on a mininet emulator and available in the Mendeley data repository [1]. In this dataset, there are three different types of attacks that have been done. The various instances present in the dataset can be divided into normal and malicious categories. The normal traffic can further be divided into TCP traffic, UDP traffic, and ICMP traffic. The malicious traffic can be further divided into TCP-SYN spoofing attacks, UDP flood spoofing attacks, and ICMP flood spoofing attacks.

Authors are motivated to work on this particular attack because this is a devastating attack and still not many significant results have been found. So, in this work, an effort is made to apply the promising Deep learning techniques to detect the DDOS attack. Deep learning techniques are more efficient and automatically detect the features which can be used to detect the attack. The contributions of the research work are described below:

Table 5.1: Comparison between Proposed and other publicly available Datasets

Dataset	***Year***	**Realistic Traffic**	***Label***	**Balanced**	***No. of Attributes***	**Format**	***Network Environment***
KDD'99	1998	No	Yes	No	41	CSV and json	traditional network
NSL-KDD	2009	Yes	Yes	No	41	ARFF	traditional network
Kyoto	2006-2009	Yes	Yes	No	24	text	traditional network
ISCX-2012	2012	Yes	Yes	No	20	PCAP	traditional network
CICIDS-2017	2017	Yes	Yes	No	83	CSV	traditional network
CSE-CIC-IDS-2018	2018	Yes	Yes	No	83	CSV	AWS
SDN-Dataset	2020	Yes	Yes	No	24	CSV	SDN

1. Various Deep learning algorithms have been evaluated for traffic classification using SDN-DDoS-Dataset. Deep learning algorithms automatically extract the significant features from the dataset which play an important role in classifying the traffic.

2. The publicly available datasets for attack detection in SDN are reviewed and shown in Table 5.1. The public datasets contain only a subset of the features which is common between traditional architecture and SDN, because of which even the best classifier is not able to attain significant classification results.

3. Furthermore, the various deep learning algorithms for traffic classification have been evaluated using publicly available datasets. The comparison of deep learning algorithms on public datasets and SDN-DDoS datasets is done. It is found that deep learning algorithms achieved better accuracy with the proposed SDN-DDoS dataset.

[1] https://data.mendeley.com/datasets/jxpfjc64kr/1

5.1.1 DDOS attack Public Datasets

Traffic classification is one of the most important areas of network management. There are broadly three approaches for network traffic classification: A port-based approach that is simple and fast but can be easily manipulated and thus not reliable. Another approach followed in literature is Deep packet inspection which provides good results but can only be used for unencrypted traffic and in the real world most of the data/traffic is encrypted. Finally, the artificial-intelligence-based approach is considered to be reliable and is the main focus of our work. The choice of the deep neural network (DNN) based model depends highly on the dataset at hand. A Convolutional Neural Network (CNN) can automatically extract traffic features from an unsecure network. The different deep learning techniques are applied to perform the task of traffic classification. Also, a comparison between different publicly available datasets is done. Some of the publicly available datasets are:

KDD'99 [88]: It is the popular dataset for network intrusion detection. The dataset was developed in 1998 and consists of 41 features. The features are grouped into three categories: general features, network features, and data-based features. There are five classes of traffic present in the dataset which includes one benign and four attack categories. The attack categories include Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R), and probe attacks. But there is a large number of duplicate records present in the dataset because of which the dataset has become obsolete.

NSL-KDD [89]: NSL-KDD is the refined form of the KDD'99 dataset. Redundant records were removed and the dataset is split into training and testing datasets. The traffic classes represented in both sets are different. Both the datasets have been in use for a long time, so they are not updated with the latest attacks. Also, the dataset is incompatible with SDN as the dataset is produced for a traditional network. Despite this, various authors have used the dataset for intrusion detection in the SDN which results in false alarms and a low rate of attack detection.

Kyoto dataset [90]: It was created at Kyoto University and consists of twenty-four features. Some of the features are in common with the NSL-KDD dataset. The normal and DDoS attack traffic is recorded simultaneously. The majority of the traffic type present in the dataset is the attack class. Also, the attack categories in the dataset are not labeled which makes it difficult in evaluating the dataset. Also, the traffic is generated in different conditions which makes it difficult to correlate the traffic instances.

ISCX2012 [91]: The dataset is created by generating the traffic via two groups. The malicious traffic is comprised of DDOS and other attacks which are created in the α group and benign traffic is created in the β group. The malicious traffic includes Denial of Service and exhaustive search attacks. It has a total of twenty features in the dataset.

Also, the normal traffic does not show the latest traffic pattern and consists of HTTP traffic instead of HTTPS traffic. The dataset is not so popular because it produces a high false alarm rate when evaluated against the machine learning algorithms.

CICIDS-2017 [92]: The dataset contains more than eighty features as compared to twenty features in the ISCX2012 dataset. Also, the normal traffic contains HTTPS traffic which provides an advantage to the dataset. But the dataset has multiple null values and missing information regarding the class labels. Furthermore, the proposed dataset size is gigantic and contains duplicate entries.

CSE-CICIDS-2018 [93]: This dataset is identical to CICIDS-2017 but deployed using Amazon (AWS) environment. The idea of using profiles to record the traffic is utilized. The dataset contains two general classes of traffic. B-profile is responsible for benign traffic generation, and M-Profile is responsible for malicious traffic generation. The traffic class of benign and malicious traffic is similar to the CICIDS-2017 dataset.

The various datasets discussed above can be compared based on the features which are available in SDN but not available in traditional Datsets. Because of this, network traffic classification research works that use machine learning approaches to classify traffic using traditional datasets are unable to identify the attack in a meaningful way. Table 5.2 depicts the various features which are present in SDN but not available in traditional network.

A point to note for all the datasets mentioned above is that they are not generated in an SDN environment. It means to measure the effectiveness of these datasets in an SDN environment might not provide accurate results. The architectural difference between both the traditional and SDN environment provides certain features which are specific to SDN and not in use by the traditional architecture. Besides, the dataset is created depending on a specific security issue. So, a compatible dataset for SDN should only be used in an SDN environment. Some of the attacks cannot be properly differentiated in traditional and SDN environments. For example, IP sweep and Port scan attacks are not considered DDoS attacks in an SDN environment, due to which the well-known classifiers cannot perform. So, using the traditional datasets in an SDN environment for intrusion detection will generate false alarms.

Table 5.2: Comparison between Proposed and other publicly available Datasets

S.No	*Feature*	Traditional Dataset	*SDN Dataset*
1	Src & Dst IP	✓	✓
2	Src & Dst Port	✓	✓
3	tx_bytes &rx_bytes	✓	✓
4	Total number of flows	✓	✓
5	Datapath	-	✓
6	Packet count per flow	-	✓
7	Byte count per flow	-	✓
8	Number of Packet_in messages	-	✓
9	Packet Rate	✓	✓
10	Src & Dst MAC Address	✓	✓
11	Ping Statistics	-	✓
12	Round trip time	-	✓

5.2 Methodology

The proposed research framework for DDOS attack detection is explained briefly. The dataset generated during the first phase of dataset generation is used in the second phase for network traffic classification. The features in the dataset are used by the deep learning algorithm for traffic classification into normal and attack classes. The various deep learning algorithms employed are listed below:

5.2.1 Deep Learning Algorithms Used

There are different ways to detect a DDoS attack. Some of them are based on the computation of some parameters, others are based on graphical methods to detect the attack, etc. In this section, the Deep learning algorithms for DDoS attack detection are discussed.

- Convolution neural network [94]: Convolution Neural network (CNN) is one of the famous models for image classification. But a variant of it, CNN-1D is applicable for text data as it is successfully applied to recommender system and various natural language processing tasks in which it yields significant performance. The important advantage of this algorithm is its ability to automatically detect important features without any human intervention.

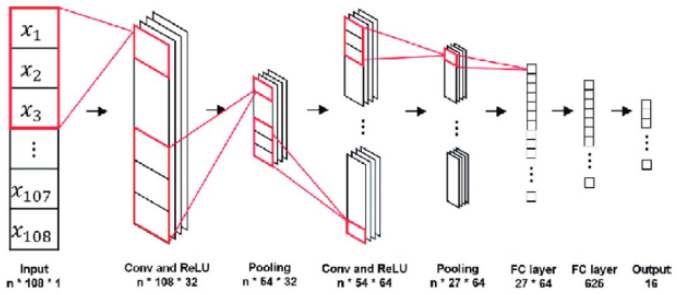

Figure 5.1: CNN-1D Architecture [1]

Figure 5.1 shows the architecture of the Convolution Neural Network when applied to one-dimensional data. When it is applied to one-dimensional data the variant of Convolution 2D is used and other layers remain the same.

- A Recurrent Neural Network [95] is best for time series data where the current state depends on previous states. As the name suggests the network is recurrent and there is a single layer that works as many layers as shown below :
  - *X (t)* means input at time t.
  - *O (t)* means output at time t.
  - *S(t)* means state at time t.
  - *W* means Weight (Constant).

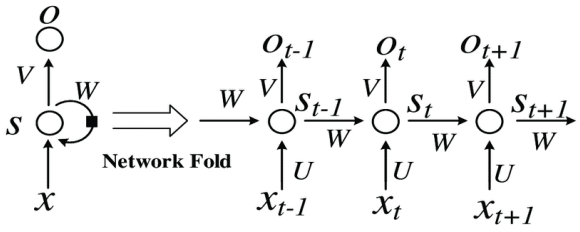

Figure 5.2: RNN Architecture [2]

Figure 5.2 represents the RNN representation, where the output at any time is a function of the present state and previous output. As the network gets deeper for RNN vanishing gradient problem arises where the model learns negligible, thus the model isn't able to reach its optimal state or exploding gradient problem arises which is vice versa, thus the model is going far from its optimal state.

- Long-Short-Term-Memory (LSTM) [95]: LSTM is a special case of RNN which is used when the gap between the past information and the place where the information is needed is small. But as the gap increases, RNN cannot work whereas LSTM does not have any such disadvantage. The problem of RNN has been solved by the LSTM architecture. LSTM architecture has been discussed as a combination of the input gate layer and tanh layer which decides what information we are going to get from the cell state and what new information we can add to the cell state respectively.

- CNN-LSTM: LSTM is the special case of RNN for solving long-range dependency. LSTM is used for long-range sequence prediction and CNN is used for feature extraction. This combination makes them suitable for various tasks including natural language processing problems where CNNs are used as feature extractors and LSTMs on audio and textual input data. The model also works well in our problem domain.

- SVC-SOM: Most of the supervised deep learning algorithms tend to have high model complexity, which results in a larger time for convergence of the models. A faster and more efficient method will aid in the intrusion detection process [55]. In this method, we use Linear SVCs along with an unsupervised deep learning method called Self Organizing Map (SOM) [96]. There is a unique SVC for each protocol type present in the data. As the dataset consists of three protocols TCP, UDP, and ICMP, there are three linear SVCs in the model.

  In our method, we have used multiple SOMs along with multiple SVCs for the classification. This resulted in better results at an expense of greater time for learning. When we tried with three different SOMs along with three different SVCs for each protocol, we found that the last method yields more accurate results at the expense of more training time.

- SAE-MLP: Stacked autoencoder [97] comprised of many autoencoders tied together where the result of one autoencoder is fed to the input of other and a SoftMax classifier is later used which is used for feature extraction.

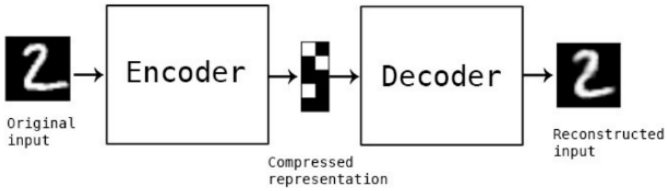

Figure 5.3: SAE [3]

Figure 5.3 shows the autoencoder in the working stage where it first encodes the input and later uses a decoder to decode which is a compressed representation of the input. An autoencoder can be thought of as a neural network but the difference between the neural network and autoencoder is that the autoencoder is comprised of two parts Encoder and Decoder resulting in a compressed representation of data than the original one. An autoencoder has a term of sparsity penalty associated with autoencoder during training. In general, stacked autoencoders outperform autoencoders in terms of model learning. Stacked Autoencoder is used in hybrid with MLP model because Multi-Layer Perceptron has the curse of dimensionality and SAE is good for dimensionality reduction. During the implementation, the use of two different optimizers is done. Stochastic gradient descent (SGD) is used for the first 10 epochs and Adam for the next 150 epochs, which reduced the training time considerably. SGD worked with a higher learning rate while Adam worked with the default initial learning rate. Thus, with a hybrid of SAE-MLP, the curse of dimensionality from MLP was lifted and it performed considerably better.

Algorithm 5 Proposed Algorithm for DDOS attack detection in SDN using Deep Learning.

Input: SDN Traffic Dataset is required as input.
Output: Classified attack Traffic from normal traffic is returned.

Initialization: Normal traffic Packet per flow, Attack traffic Packet per flow, Count of packet_in messages generated, Count of flows generated, Packet Rate.

1: For each flow, Deep Learning Algorithm analyses the features present in the dataset. Dataset is analyzed and processed by following steps.
2: **for** $i = flow_1$ to $flow_n$ **do**
3: To collect the flow statistics OFPFlowStatsRequest method is invoked and the required flow statistics are invoked from the switch at a regular interval.
4: To collect the port statistics OFPPortStatsRequest method is invoked and the the required port statistics are invoked from the switch at a regular interval.
5: $Src\_IP \rightarrow$ Digit_Extract(Src_IP)
6: $Dst\_IP \rightarrow$ Digit_Extract(Dst_IP)
7: $SwitchID \rightarrow$ Sort($SwitchID$)
8: $flow_{i=1} \rightarrow$ Encode($flow_{i=n}$)
9: $flow_{i=1} \rightarrow$ Normalize($flow_{i=n}$)
10: Split the dataset into Train and Test dataset.
11: The significant features are selected using the Recursive feature elimination method.
12: ($features_n$)$\rightarrow$ ($features_{n-15}$)
13: Train data$\xrightarrow{\text{transform}}$SF Train Dataset
14: Test data$\xrightarrow{\text{transform}}$SF Test Dataset
15: Deep Learning Model $\xrightarrow{\text{SF Train Dataset}}$SF Trained Model.
16: Evaluate the performance
SF Trained Model$\xrightarrow{\text{evaluate}}$SF Test Dataset. Deep Learning Model $\rightarrow$ Tuned (Deep learning model)
17: Repeat and Go to Step 3.
18: **end for**
19: **return** Attack traffic classified from benign traffic.

5.2.2 Proposed Methodology

In this section, the method which has been followed in the work is discussed. We also show the step-by-step procedure of methodology given below. The steps taken during the methodology are depicted in Algorithm 5. The module DigitExtract() extracts the last digit from the features SrcIP and DstIP. Sort() is the module that sorts the dataset by the switch ID field. Encode() takes the flow information and encodes the categorical features in the dataset into the vector form required by machine learning algorithms. Because some features in the dataset have small values while others have large values, machine learning algorithms may produce biased results.

The Normalize() module normalizes the features in the dataset to a value between 0 and 1. Using feature selection methods, the important features in the dataset can be identified. There

are various feature selection methods available in the literature, but when applied to our dataset, they all produce mostly the same results, so we used the Recursive Elimination feature selection method to select the significant features present in the dataset. The various Deep Learning models mentioned are trained on the transformed dataset, which is the updated dataset after non-significant features are removed. After the Deep Learning models have been evaluated on the test dataset, their performance is measured using metrics such as Accuracy and F1- Score. Tuning the hyperparameters improves the model's performance where hyperparameters are tabulated above. Finally, with a significant accuracy score, the optimized model classifies the traffic dataset into normal and attack classes.

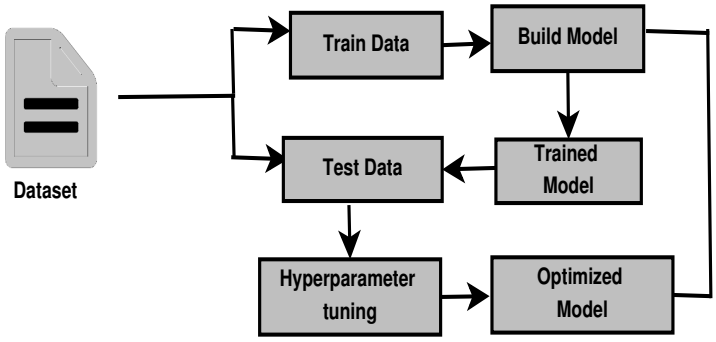

Figure 5.4: Block diagram of the methodology followed.

Figure 5.4 shows the methodology which has been followed to for traffic classification into attack and benign classes. The different steps involved for the same are discussed below:

1. Data Pre-Processing: The dataset consists of 22 features and is available in the Mendeley data repository. The dataset's features are mostly statistical in nature. The traffic flow in our dataset contains three different protocols: TCP, UDP, and ICMP. The dataset is annotated programmatically by labeling the traffic by setting the variable, which differentiates the various traffic categories. The label denotes whether the traffic is an attack or normal. The dataset consists of features such as source and destination IP addresses along with providing information about the bytes, packets, duration, transfer speed, etc. Dataset pre-processing involves removing the redundant flows in the dataset and other attributes which are zero. The next step is to identify the type of the variables i.e., whether the variable is categorical or numerical, and encode the categorical variables. Here, categorical variables such as Source IP, Destination IP, Switch ID, and Protocol are encoded. All the categorical variables were encoded by using one-hot-encoding. After that, the normalization [98] of the features by using min-max scalar is done. The values in the dataset

are normalized within the range of 0-1. The next step is to find the correlation between the various features. The correlation score of various features was compared with the threshold and if it is found to be greater than the threshold, the particular feature is a less significant one and it needs to be removed. The significant features are fed to the deep learning model.

2. Apply Deep Learning classifiers: In this step, we apply various Deep learning classifiers which have been discussed in section 5.2.1. The dataset is trained on a particular model and then it is tested on the unseen data.

3. Evaluation: In this step, the applied models were evaluated using measures of accuracy, precision, recall, F-score, False positive rate, and False negative rate.

4. Hyperparameter Tuning: After the evaluation of the dataset is done, further improvements can be performed in the model performance through hyperparameter tuning. Various hyperparameters that have been used are shown in Table 5.3 and include the number of layers, the number of epoch and the regularization parameter which when tuned give optimized results.

Table 5.3: Parameters used in SAE-MLP model

S.No.	***Parameter***	***Value***
1	Filters at Convolution layer.	64
2	kernel size	3
3	Activation function at autoencoder.	sigmoid
4	Padding at decoder.	1
5	Activation function at MLP	relu
6	hidden nodes in the MLP layer	128
7	Activation function at MLP layer	tanh
8	Batch size	64
8	Learning rate	0.001
9	Optimization function	adam
10	Epochs	50
11	Optimization function	SGD

5.3 Implementation Details and Results

In this section, the implementation environment utilized for performing the experiments has been discussed. The work has been carried out on HP EliteBook with 16 GB RAM and 64-bit processor on windows 10. A single Ryu controller is connected to the open-vswitch and the vswitch is connected to the various hosts. Benign traffic is generated from random hosts with the help of mgen tool at the rate of 450 packets per second. The benign and malicious traffic are generated with a specific packet size for a specific duration. The work makes use of the dataset generated for DDOS attack detection in SDN.

To evaluate the performance of different models, some parameters are measured and evaluated. Evaluation parameters include Accuracy, Precision, Recall, F1-score and are discussed through the Confusion matrix. Apart from the above-mentioned evaluators some of them are also mentioned below:

False Positive rate(FPR): It can be interpreted as the false alarms raised due to incorrect predictions of the attack class to be normal class.

$$FPR = (f_p/(t_n + f_p)) \tag{5.1}$$

FPR must be less to keep the false alarms nil and should be minimized.

False Negative Rate (FNR): It can be interpreted as the false alarms raised due to incorrect predictions of the normal class to attack class.

$$FNR = (f_n/(t_p + f_n)) \tag{5.2}$$

Here, t_p, t_n, f_p, f_n are the True Positives, True Negatives, False Positives, False Negatives and are computed by using the dataset. It can also be computed as one minus true positive rate. From Table 5.4, it is clear that the accuracy achieved by the proposed SAE-MLP model achieved the best results. The SDN-dataset [52] utilized has an imbalanced count of classes, so other evaluators including Precision, recall, and F1-score need to be measured where a large value of F1-score indicates promising results.

Table 5.4: Classifier Performances

Model	*Acc*	NPrec	*APrec*	NRecall	*ARecall*	NFScore	*AFScore*	FPR	*FNR*
CNN	0.9874	0.9875	0.9873	0.9890	0.9855	0.9883	0.9864	0.0144	0.0109
LSTM	0.9560	0.9620	0.9490	0.9564	0.9556	0.9592	0.9523	0.0443	0.0435
CNN-LSTM	0.9948	0.9943	0.9955	0.9966	0.9926	0.9954	0.9940	0.0073	0.0033
SVC-SOM	0.9545	0.9671	0.9375	0.9540	0.9551	0.9605	0.9462	0.0448	0.0459
SAE-MLP	0.9975	0.9996	0.9969	0.9977	0.9994	0.9987	0.9982	0.0005	0.0022

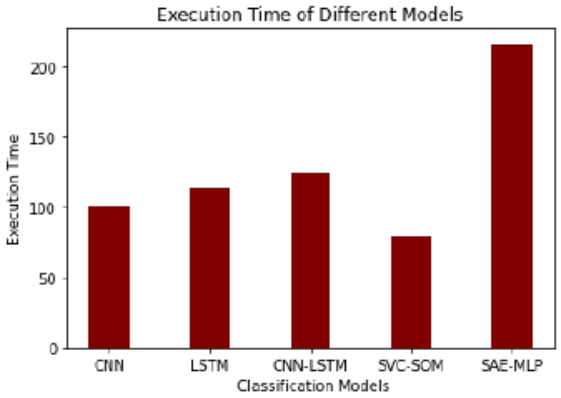

Figure 5.5: Execution Time of Different Models

5.3.1 Experimental Results and Analysis

In this section, various results attained are discussed. The performance of different deep learning models is evaluated against the different performance parameters. From Table 5.4, it is clear that the accuracy achieved by the proposed SAE-MLP model achieved the best results. Table 5.5 compares the execution times of different algorithms. It demonstrates that the SAE-MLP model takes the longest to execute when compared to other models. The highest accuracy of 99.75% achieved by the SAE-MLP model comes at the cost of a 216.39 seconds execution time. Figure 5.5 depicts a pictorial representation as well.

Table 5.5: Execution Time required by different ML Algorithm

S.No.	*Algorithm*	*Execution Time in seconds*
1	CNN	101.54
2	LSTM	113.21
3	CNN-LSTM	124.32
4	SVC-SOM	79.58
5	SAE-MLP	216.39

Table 5.6: Performance of different classifiers on public datasets

Dataset	Model	*Acc*	NPrec	*APrec*	NRecall	*ARecall*	NFScore	*AFScore*	FPR	*FNR*
KDD'19	CNN	0.8874	0.8875	0.8873	0.8890	0.8855	0.8883	0.8864	0.1144	0.1109
	LSTM	0.7760	0.7420	0.7490	0.7564	0.7556	0.7592	0.7523	0.0443	0.0435
	CNN-LSTM	0.9848	0.9875	0.9855	0.9866	0.9826	0.9854	0.9840	0.0073	0.0033
	SVC-SOM	0.9545	0.9671	0.9375	0.9540	0.9551	0.9605	0.9462	0.0448	0.0459
	SAE-MLP	0.9835	0.9996	0.9969	0.9977	0.9994	0.9987	0.9982	0.0005	0.0022
NSLKDD	CNN	0.9874	0.9875	0.9873	0.9890	0.9855	0.9883	0.9864	0.0144	0.0109
	LSTM	0.9560	0.9620	0.9490	0.9564	0.9556	0.9592	0.9523	0.0443	0.0435
	CNN-LSTM	0.9948	0.9943	0.9955	0.9966	0.9926	0.9954	0.9940	0.0073	0.0033
	SVC-SOM	0.9545	0.9671	0.9375	0.9540	0.9551	0.9605	0.9462	0.0448	0.0459
	SAE-MLP	0.9953	0.9986	0.9959	0.9957	0.9964	0.9947	0.9981	0.0006	0.0034
kyoto	CNN	0.9765	0.9675	0.9573	0.9459	0.9765	0.9876	0.9567	0.0543	0.0238
	LSTM	0.9560	0.9620	0.9490	0.9564	0.9556	0.9592	0.9523	0.0443	0.0435
	CNN-LSTM	0.9848	0.9743	0.9455	0.9676	0.9852	0.9654	0.9784	0.0673	0.0784
	SVC-SOM	0.9785	0.9171	0.9475	0.9340	0.9231	0.9405	0.9532	0.0231	0.0769
	SAE-MLP	0.9785	0.9679	0.9623	0.9777	0.9594	0.9687	0.9638	0.0021	0.0543
ISCX-2012	CNN	0.9754	0.9865	0.9785	0.9860	0.9812	0.9683	0.9764	0.0245	0.0342
	LSTM	0.9760	0.9650	0.9790	0.9464	0.9856	0.9792	0.9823	0.0454	0.0364
	CNN-LSTM	0.9485	0.9643	0.9578	0.9678	0.9756	0.9854	0.9780	0.0065	0.0073
	SVC-SOM	0.9645	0.9643	0.9575	0.9340	0.9561	0.9805	0.9762	0.0573	0.0684
	SAE-MLP	0.9891	0.9896	0.9869	0.9856	0.9894	0.9865	0.9782	0.0785	0.0673
CICIDS-2017	CNN	0.9674	0.9775	0.9573	0.9864	0.9675	0.9783	0.964	0.0144	0.0109
	LSTM	0.9560	0.9620	0.9490	0.9564	0.9556	0.9592	0.9523	0.0443	0.0435
	CNN-LSTM	0.9948	0.9943	0.9955	0.9966	0.9926	0.9954	0.9940	0.0073	0.0033
	SVC-SOM	0.9545	0.9671	0.9375	0.9540	0.9551	0.9605	0.9462	0.0448	0.0459
	SAE-MLP	0.9745	0.9996	0.9969	0.9977	0.9994	0.9987	0.9982	0.0005	0.0022
CSE-CICIDS-2018	CNN	0.9543	0.9675	0.9763	0.9564	0.9674	0.9754	0.9676	0.03238	0.609
	LSTM	0.9560	0.9620	0.9490	0.9564	0.9556	0.9592	0.9523	0.0443	0.0435
	CNN-LSTM	0.9448	0.9243	0.9555	0.9666	0.9743	0.9654	0.9567	0.0015	0.0033
	SVC-SOM	0.9545	0.9671	0.9375	0.9540	0.9551	0.9605	0.9462	0.0448	0.0459
	SAE-MLP	0.9775	0.9796	0.9469	0.9677	0.9894	0.9787	0.9582	0.0007	0.0042
SDN-Dataset-2020	CNN	0.9874	0.9875	0.9873	0.9890	0.9855	0.9883	0.9864	0.0144	0.0109
	LSTM	0.9560	0.9620	0.9490	0.9564	0.9556	0.9592	0.9523	0.0443	0.0435
	CNN-LSTM	0.9948	0.9943	0.9955	0.9966	0.9926	0.9954	0.9940	0.0073	0.0033
	SVC-SOM	0.9545	0.9671	0.9375	0.9540	0.9551	0.9605	0.9462	0.0448	0.0459
	SAE-MLP	0.9975	0.9996	0.9969	0.9977	0.9994	0.9987	0.9982	0.0005	0.0022

Table 5.6 shows the performance of the different deep learning algorithms on various public datasets discussed above. The different datasets contain an unequal and varying count of attack classes present. These datasets are used by researchers for attack detection but are not generated for SDN and are rather created for the traditional networks.

The datasets created for the traditional network have certain features which are not applicable in an SDN environment. Data exploration is done to understand the data, the number of instances in each class of traffic, Normal and malicious traffic distribution in the dataset. This summary data helps to comprehend the dataset. Figure 5.6 shows the distribution of the benign traffic in the dataset. Similarly, the distribution of attack traffic in the dataset is depicted in Figure 5.7. From the Figure, it can be inferred that the dataset is not biased towards a particular class and it is a balanced dataset.

The different performance parameters are also calculated and comparative evaluation results

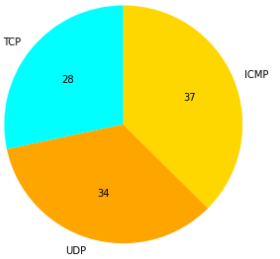

Figure 5.6: Percentage of Normal traffic present in the Proposed dataset.

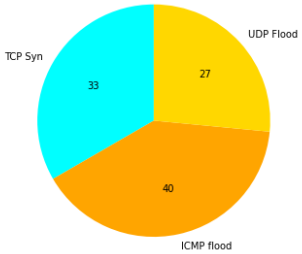

Figure 5.7: Percentage of Attack Traffic present in the Proposed dataset.

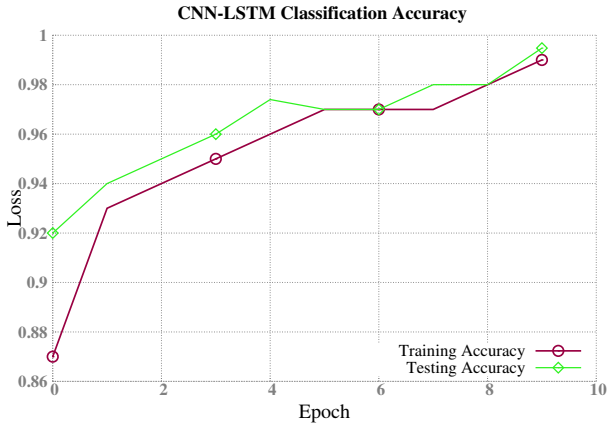

Figure 5.8: Classification Accuracy achieved with CNN-LSTM Model.

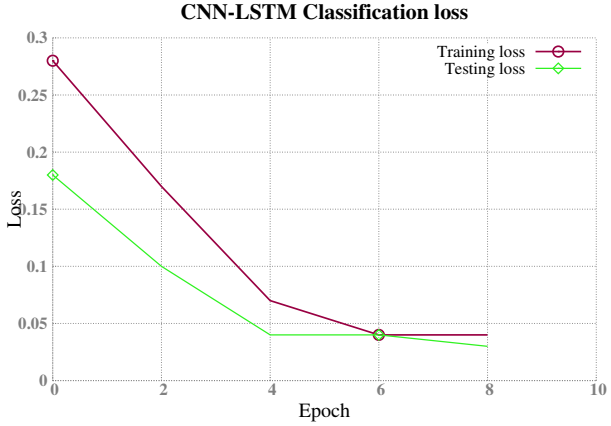

Figure 5.9: Classification loss incurred with the CNN-LSTM model.

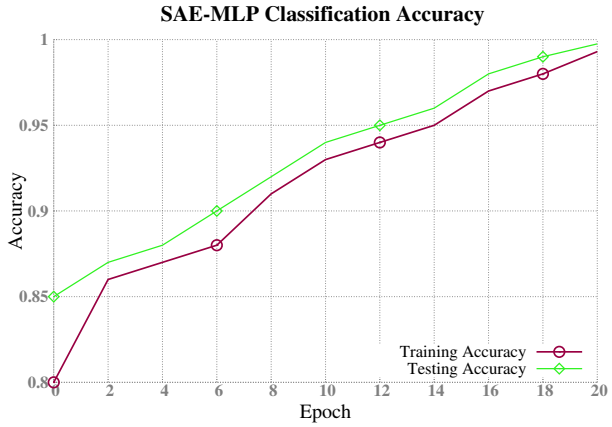

Figure 5.10: Classification Accuracy achieved with SAE-MLP Model.

Figure 5.11: SAE-MLP Reconstruction loss

are presented. Figure 5.8 depicts the accuracy score of 98.8% achieved by the CNN-LSTM model plotted against the number of epoch. The hybrid CNN-LSTM model has achieved significant accuracy as the hybrid of CNN which detects the significant features and LSTM which predicts the class label worked well after amalgamation. Figure 5.9 shows the cross-entropy loss decreases with an increase in the epoch. It shows that as the number of times the model is trained on the dataset, the error in classifying the traffic decreases.

Figure 5.10 shows that the accuracy score of 99.75% is achieved by the hybrid Stacked autoencoder-Multi-layer-Perceptron model (SAE-MLP) which used the MLP model in combination with SAE and attains the highest accuracy. The MLP model suffers from the curse-of-dimensionality problem and autoencoders work to reduce the dimensionality problem. Figure 5.11 shows the SAE Reconstruction loss where the loss in reconstructing the input image has reduced within 160 epochs with the use of two different optimizers. Stochastic gradient Descent and Adam optimizers are used for 10 and 150 epochs respectively.

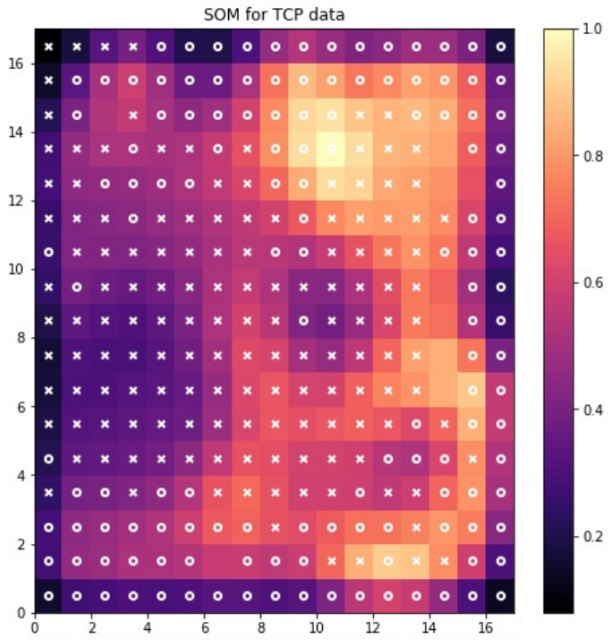

Figure 5.12: SOM Plotted for TCP data out of the complete dataset.

Figure 5.12 shows the SOM plotted for the TCP data of our dataset. The color bar shows the mean inter-neuron distance where the higher value indicates a homogenous cluster and lower values indicate the less homogenous cluster. 'x' value represents the malicious traffic and 'o' value represents the benign traffic. At first, the Linear SVCs classify the flow entries from OpenFlow switches. The Linear Support Vector Classifier learns a linear decision boundary accompanied by two margin lines equidistant from the decision boundary, to classify points from two classes. However, depending upon the separating hyperplane and the width of the margin learned by the SVC, some data points may lie within the two margins. These data points can be misclassified as the region is allowed to encompass points from both classes. So for the data points lying within the two margins, called suspicious points, inference from SOM is used for the final classification. When we tried three different SOMs along with three different SVCs for each protocol, we found that the SAE-MLP yields more accurate results at the cost of a greater training time. But SOM attains less accuracy score of 95.45% as compared to the supervised algorithm of SAE-MLP which attains the highest accuracy with the SDN dataset. This may be due to the unsupervised nature of the algorithm used.

It is similar to clustering algorithms popular in data mining literature, although the method employed is very different. The clusters learned by the SOM also depend upon the neighboring clusters(neurons) in the map. This neighborhood is determined by the initial radius set for the SOM, which decays over time. The training process should be stopped when changes made

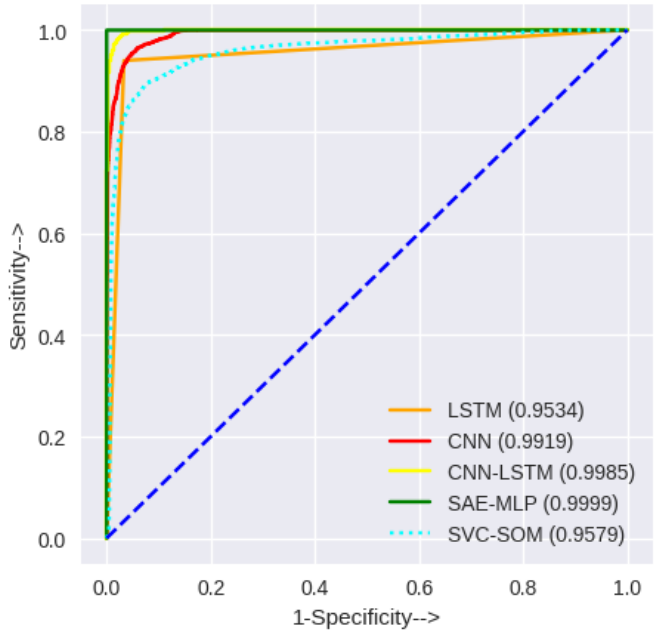

Figure 5.13: ROC Curve Plotted for the different ML algorithms.

in the map start becoming minimal or insignificant. There are deeper details relating to the various hyperparameters employed. Figure 5.13 depicts the Receiver Operating Curve (ROC) of various algorithms used to detect a DDoS attack. The area under the curve (AUC) demonstrates the classifier's ability to differentiate between different classes. The greater the AUC value, the better the classifier's ability to distinguish between positive and negative classes. The AUC value of 99.99 for SAE-MLP indicates the classifier's significance performance. The SAE-MLP classifier employed for classification plays a significant role in attack detection as the hybrid of SAE-MLP model attains significant accuracy.

The proposed dataset can be evaluated for other ML algorithms as the dataset is publicly available. But the hybrid of SAE and MLP achieves significant results with the chosen hyper-parameters.

The proposed approach is compared to existing research work in Table 5.7. The existing highest result shows an accuracy score of 99.10% and the SAE-MLP model with the proposed dataset shows an accuracy score of 99.73%. The SAE-MLP classifier followed plays a significant role in attack detection as the hybrid of SAE and MLP model attains significant accuracy.

Table 5.7: Proposed work vs State-of-the-art Research

S.No	Authors	*Testing Accuracy*
1	Wang et al. [1]	99.10%
2	Agarwal et al. [99]	95.35%
3	Niyaz et al. [18]	95.65%
4	Phan et al. [68]	97.6%
5	Abdulqadder et al. [100]	80%
6	Sambangi et al. [101]	97.86%
7	**Proposed**	**99.73%**

5.3.2 Observation and Discussion

In this segment, we will critically analyze the performance of the results obtained. The dataset is created using the SDN emulator 'mininet' and is available online. In our previous work, different machine learning algorithms have been trained on the dataset, and a classification of the traffic has been performed. Out of the various machine learning classifiers used, the hybrid model of Support vector Classifier with Random Forest achieved the highest accuracy of 98.8% in traffic classification [39]. Although a good classification accuracy has been achieved with machine learning algorithms, many deep learning algorithms were showing promising results in this area of traffic classification. So, the dataset is trained with different deep learning algorithms in the proposed work.

Various supervised and unsupervised learning algorithms have been trained and tested on the given dataset. But unsupervised algorithms show relatively low classification performance as compared to supervised algorithms. This may be due to the training of the unsupervised learning algorithm leads to underfitting and does not provide significant results. As can be seen, the SAE-MLP model achieves significant results as MLP which is a supervised algorithm is trained on the given dataset which is later fed to SAE which is a form of self-supervised learning algorithm for dimensionality reduction and thus optimized well on the classification task. On the other hand, Self Organizing Maps(SOM) is an unsupervised learning algorithm that cluster the data instances based on the neighbor clusters. The area covered by neighbors is determined by the initial radius which decays over time. As the proposed work is done using an experimental dataset that is emulated on the mininet emulator, the trained deep learning model can be applied in real-time to detect the attack with approximately 100 percent accuracy. Previously, attack detection is done by using statistical approaches, which involve calculation of the various parameters, and

depending on the value of parameters, the attack detection is performed.

5.4 Summary

SDN emerged as a revolutionary network for the future which relates to the flexibility requirement of the future network. In this chapter, we have extended the work of DDoS attack detection with Deep Learning algorithms. The public datasets for traditional networks contain only a subset of the features which is common between traditional architecture and SDN, because of which even the best classifier is not able to attain significant classification results. Among the various algorithms used, the hybrid model of Stacked AutoEncoder with Multi-layer perceptron achieved the higest detection accuracy of 99.73%.

Some facts about the different security attacks have been captured in previous chapters. Based on these facts, a few conclusions are drawn in the following chapter. The following chapter also suggests future work for the reader.

CHAPTER 6
CONCLUSION AND FUTURE WORK

SDN provides an exclusive chance to pursue the dynamic network requirements while also delivering the best, customizable, and dependable solutions. A secure network is required to achieve this more effectively. The preceding chapters present novel solutions for detecting DDoS and ARP-based attacks. This chapter will provide concluding remarks and future directions for researchers in the domain of SDN Security.

SDN (software-defined networking) has revolutionized the networking industry and completely transformed the networking domain. In a traditional network, the devices are configured manually, which is error-prone and inflexible for larger networks. SDN assists in programming the network and thus automatically configuring the network. However, in addition to the numerous benefits provided, SDN is vulnerable to various attacks due to the centralized architecture.

Network attacks occurring in SDN could indeed be similar to traditional network attacks such as DDoS attacks. But the way of detecting and preventing such attacks becomes different in SDN. For effective attack detection, an application needs to be written thatdeveloped in a traditional network environment checks for the attack condition. The previous work done for DDoS attack detection has not yielded significant results and remains a challenge. Secure network communication is difficult to imagine without reliable data link layer host identification process. ARP-based host discovery is carried out at the data link layer. Attacks such as ARP Poisoning and Flooding are possible. The literature on secure communication between hosts is deficient. Various gaps in the literature about these attacks are identified, i.e., insecure, if secure, then costly in terms of computation and processor load.

The thesis makes five major contributions: Creating an SDN-specific DDoS attack Dataset, Creating an ARP-Poisoning and ARP-Flood Attack Dataset, Detection of DDoS attack by applying Deep Learning Techniques and, Traffic classification into ARP Poison, Flood attack, and benign traffic classes. Few conclusions can be drawn from each such contribution.

1. Previous research did not address the development of SDN-specific datasets. The author addressed the issue and created an SDN-specific DDoS dataset for network traffic classification. The public datasets available for traffic classification in SDN are either not available for free download or are developed in traditional network environment.

2. Since there has been very little research into ARP-based attack datasets, the development of datasets is required for research in the SDN environment. Dataset is also created for ARP Poison and flood attack and is utilized by machine learning algorithm for traffic classification.

3. The results attained in detecting DDoS attack was found to be best with a hybrid model of Support Vector Machine with Random Forest. The previous research in DDoS attack detection is either based on statistical technique or pattern matching but ML technique offers a promising solution.

4. The detection of DDoS attacks has been improved by using Deep Learning algorithms, which yielded higher accuracy. Deep Learning algorithms automatically extract the significant features present in the dataset, which can aid in more accurately classifying the attack traffic.

5. The hybrid model of Convolution Neural Network with Longest Short Term Memory (CNN-LSTM) was found to have a significant detection accuracy in detecting ARP Poison attack and ARP Flood attack. ARP-based attacks are also common in SDN and can result in MITM and DoS attacks. The author also used machine learning techniques to detect the ARP-Poison and Flood attacks, with promising results.

6.1 Future Scope

After dedicating a considerable amount of time collecting statistics and making judgments in any research, there is always room for improvement. In addition, the thesis suggests a few future directions for making host discovery and resiliency to DDoS, ARP-based attacks. Some of the thesis's future scope are discussed below:

1. Other attacks related to Topology Poisoning attacks which include Link Fabrication Attack, Link relay attack, and Host Location Hijacking attack can also be undertaken by future researchers. The SDN- specific dataset can be created for each of these attacks and network traffic classification can be done with state-of-the-art algorithms.

2. On current deployments, several attacks such as DDoS, ARP Poison, and ARP Flood are evaluated using Ryu Controller. There are many other controllers in SDN such as POX, OpenDayLight, Floodlight, and other SDN controllers ONOS, Beacon, and HPE-VAN. Each of the aforementioned controllers can be assessed for vulnerability to one or more of the aforementioned attacks.

3. The controller assumes that all knowledge gained is legitimate. If the controller is malicious, it is far more difficult to stop, diagnose, or lessen ARP Poison, Flooding, and DDoS attacks. The detection of such a case becomes interesting.

4. Flow statistics, port statistics, and other flow parameters, are scheduled tasks in research to create the dataset. Experiments are still needed to determine the best time to collect statistics. Selecting the best time will reduce overhead and end up making the network less vulnerable to malicious activities.

5. A generalized dataset could be generated that tackles all kinds of possible attacks in SDN networks.

REFERENCES

[1] Wei Wang, Yiqiang Sheng, Jinlin Wang, Xuewen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE access*, 6:1792–1806, 2017.

[2] Anderson Santos Da Silva, Juliano Araujo Wickboldt, Lisandro Zambenedetti Granville, and Alberto Schaeffer-Filho. atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn. In *IEEE 2016, Proc. of Network Operations and Management Symposium (NOMS)*, pages 27–35, 2016.

[3] Chaitanya Buragohain and Nabajyoti Medhi. flowtrapp: An sdn based architecture for ddos attack detection and mitigation in data centers. In *IEEE 2016, Proc. of 3rd International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 519–524, 2016.

[4] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Software-defined networking security: pros and cons. *IEEE Communications Magazine*, 53(6):73–79, 2015.

[5] Jaouad Benabbou, Khalid Elbaamrani, and Noureddine Idboufker. Security in openflow-based sdn, opportunities and challenges. *Photonic Network Communications*, 37(1):1–23, 2019.

[6] Neelam Dayal and Shashank Srivastava. an rbf-pso based approach for early detection of ddos attacks in sdn. In *IEEE 2018, Proc. of 10th International Conference on Communication Systems & Networks (COMSNETS)*, pages 17–24, 2018.

[7] Zakaria Abou El Houda, Lyes Khoukhi, and Abdelhakim Senhaji Hafid. Bringing intelligence to software defined networks: mitigating ddos attacks. *IEEE Transactions on Network and Service Management*, 17(4):2523–2535, 2020.

[8] Neha Agrawal and Shashikala Tapaswi. An sdn-assisted defense mechanism for the shrew ddos attack in a cloud computing environment. *Journal of Network and Systems Management*, 29(2):1–28, 2021.

[9] Cheng-Yu Cheng, Edward Colbert, and Hang Liu. experimental study on the detectability of man-in-the-middle attacks for cloud applications. In *IEEE 2019, Proc. of Cloud Summit*, pages 52–57, 2019.

[10] Husain Abdulla, Hamed Al-Raweshidy, and Wasan S Awad. Arp spoofing detection for iot networks using neural networks. 2020.

[11] Shi Dong and Mudar Sarem. Ddos attack detection method based on improved knn with the degree of ddos attack in software-defined networks. *IEEE Access*, 8:5039–5048, 2019.

[12] Hannah AS Adjei, Mr Tan Shunhua, George K Agordzo, Yangyang Li, Gregory Peprah, and Emmanuel SA Gyarteng. Ssl stripping technique (dhcp snooping and arp spoofing inspection). In *IEEE 2021 Proc. of 23rd International Conference on Advanced Communication Technology (ICACT)*, pages 187–193, 2021.

[13] Kübra Kalkan, Levent Altay, Gürkan Gür, and Fatih Alagöz. Jess: Joint entropy-based ddos defense scheme in sdn. *IEEE Journal on Selected Areas in Communications*, 36(10):2358–2372, 2018.

[14] Bing Wang, Yao Zheng, Wenjing Lou, and Y Thomas Hou. Ddos attack protection in the era of cloud computing and software-defined networking. *Computer Networks*, 81:308–319, 2015.

[15] Nisharani Meti, DG Narayan, and VP Baligar. detection of distributed denial of service attacks using machine learning algorithms in software defined networks. In *IEEE 2017,*

Proc. of International conference on advances in computing, communications and informatics (ICACCI), pages 1366–1371, 2017.

[16] Hongyu Liu, Bo Lang, Ming Liu, and Hanbing Yan. Cnn and rnn based payload classification methods for attack detection. *Knowledge-Based Systems*, 163:332–341, 2019.

[17] Majd Latah and Levent Toker. An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks. *CCF Transactions on Networking*, 3(3):261–271, 2020.

[18] Quamar Niyaz, Weiqing Sun, and Ahmad Y Javaid. A deep learning based ddos detection system in software-defined networking (sdn). *arXiv preprint arXiv:1611.07400*, 2016.

[19] Narayanan Anand, Sarath Babu, and BS Manoj. On detecting compromised controller in software defined networks. *Computer Networks*, 137:107–118, 2018.

[20] Ajay Nehra, Meenakshi Tripathi, and Manoj Singh Gaur. ficur: Employing sdn programmability to secure arp. In *IEEE 2017, Proc. of 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1–8, 2017.

[21] Mauro Conti, Nicola Dragoni, and Viktor Lesyk. A survey of man in the middle attacks. *IEEE Communications Surveys & Tutorials*, 18(3):2027–2051, 2016.

[22] Sungmin Hong, Lei Xu, Haopei Wang, and Guofei Gu. Poisoning network visibility in software-defined networks: New attacks and countermeasures. In *NDSS*, volume 15, pages 8–11, 2015.

[23] Seung Yeob Nam, Dongwon Kim, and Jeongeun Kim. Enhanced arp: preventing arp poisoning-based man-in-the-middle attacks. *IEEE communications letters*, 14(2):187–189, 2010 .

[24] M Carnut and J Gondim. arp spoofing detection on switched ethernet networks: A feasibility study. In *Proc. of 5th Simposio Seguranca em Informatica*, 2003.

[25] Ahmed M AbdelSalam, Ashraf B El-Sisi, and Vamshi Reddy. Mitigating arp spoofing attacks in software-defined networks. *ICCTA 2015, At Alexandria, Egypt*, 2015.

[26] Shuhua Deng, Xing Gao, Zebin Lu, and Xieping Gao. Packet injection attack and its defense in software-defined networks. *IEEE Transactions on Information Forensics and Security*, 13(3):695–705, 2017.

[27] Kai Zhang and Xiaofeng Qiu. cmd: A convincing mechanism for mitm detection in sdn. In *IEEE 2018, Proc. of International Conference on Consumer Electronics (ICCE)*, pages 1–6, 2018.

[28] Enrique De la Hoz, Gary Cochrane, Jose Manuel Moreira-Lemus, Rafael Paez-Reyes, Ivan Marsa-Maestre, and Bernardo Alarcos. detecting and defeating advanced man-in-the-middle attacks against tls. In *IEEE 2014, Proc. of 6th International Conference On Cyber Conflict (CyCon 2014)*, pages 209–221, 2014.

[29] Xin Wang, Neng Gao, Lingchen Zhang, Zongbin Liu, and Lei Wang. novel mitm attacks on security protocols in sdn: A feasibility study. In *Springer, Proc. of International Conference on Information and Communications Security*, pages 455–465, 2016.

[30] Rogério Leão Santos De Oliveira, Christiane Marie Schweitzer, Ailton Akira Shinoda, and Ligia Rodrigues Prete. Using mininet for emulation and prototyping software-defined networks. In *IEEE 2014, Proc. of Colombian Conference on Communications and Computing (COLCOM)*, pages 1–6, 2014.

[31] Seungwon Shin and Guofei Gu. attacking software-defined networks: A first feasibility study. In *Proc. of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 165–166, 2013.

[32] Michael Brooks and Baijian Yang. a man-in-the-middle attack against opendaylight sdn controller. In *Proc. of the Annual 4th ACM Conference on Research in Information Technology*, pages 45–49, 2015.

[33] Octopress. Mininet emulation Software. http://www.mininet.org/, 2018. [Online; accessed 19-July-2020].

[34] Sriram Natarajan. Ryu Controller. http://www.sdnhub.org/tutorials/ryu/, 2014. [Online; accessed 19-March-2021].

[35] Nagarathna Ravi, S Mercy Shalinie, and D Danyson Jose Theres. Balance: Link flooding attack detection and mitigation via hybrid-sdn. *IEEE Transactions on Network and Service Management*, 17(3):1715–1729, 2020.

[36] Anass Sebbar, Mohammed Boulmalf, Mohamed Dafir Ech-Cherif El Kettani, and Youssef Baddi. detection mitm attack in multi-sdn controller. In *IEEE 2018, Proc. of 5th International Congress on Information Science and Technology (CiSt)*, pages 583–587, 2018.

[37] Weverton Luis da Costa Cordeiro, Jonatas Adilson Marques, and Luciano Paschoal Gaspary. Data plane programmability beyond openflow: Opportunities and challenges for network and service operations and management. *Journal of Network and Systems Management*, 25(4):784–818, 2017.

[38] Nagarathna Ravi and S Mercy Shalinie. Learning-driven detection and mitigation of ddos attack in iot via sdn-cloud architecture. *IEEE Internet of Things Journal*, 7(4):3559–3570, 2020.

[39] Nisha Ahuja, Gaurav Singal, Debajyoti Mukhopadhyay, and Neeraj Kumar. Automated ddos attack detection in software defined networking. *Journal of Network and Computer Applications*, page 103108, 2021.

[40] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues. Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in sdn: A social multimedia perspective. *IEEE Transactions on Multimedia*, 21(3):566–578, 2019.

[41] Haopei Wang, Lei Xu, and Guofei Gu. floodguard: A dos attack prevention extension in software-defined networks. In *IEEE 2015, Annual Proc. of 45th International Conference on Dependable Systems and Networks*, pages 239–250, 2015.

[42] Changhoon Yoon, Seungsoo Lee, Heedo Kang, Taejune Park, Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. Flow wars: Systemizing the attack surface and defenses in software-defined networks. *IEEE/ACM Transactions on Networking*, 25(6):3514–3530, 2017.

[43] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Machine-learning techniques for detecting attacks in sdn using nsl-kdd. *arXiv preprint arXiv:1910.00817*, 2019.

[44] Hong Kiwon et al. Sdn-assisted slow http ddos at tack defense method. *IEEE Communications Letters*, 20(17):1–1, 2017.

[45] Afsaneh Banitalebi Dehkordi, MohammadReza Soltanaghaei, and Farsad Zamani Boroujeni. The ddos attacks detection through machine learning and statistical methods in sdn. *The Journal of Supercomputing*, 77(3):2383–2415, 2021.

[46] Francesco Palmieri. Network anomaly detection based on logistic regression of nonlinear chaotic invariants. *Journal of Network and Computer Applications*, 148:102460, 2019.

[47] Reneilson Santos, Danilo Souza, Walter Santo, Admilson Ribeiro, and Edward Moreno. Machine learning algorithms to detect ddos attacks in sdn. *Concurrency and Computation: Practice and Experience*, page e5402.

[48] Myo Myint Oo, Sinchai Kamolphiwong, Thossaporn Kamolphiwong, and Sangsuree Vasupongayya. Advanced support vector machine-(asvm-) based detection for distributed denial of service (ddos) attack on software defined networking (sdn). *Journal of Computer Networks and Communications*, 2019, 2019.

[49] Yunhe Cui, Lianshan Yan, Saifei Li, Huanlai Xing, Wei Pan, Jian Zhu, and Xiaoyang Zheng. Sd-anti-ddos: Fast and efficient ddos defense in software-defined networks. *Journal of Network and Computer Applications*, 68:65–79, 2016.

[50] Nisha Ahuja, Gaurav Singal, and Debajyoti Mukhopadhyay. dlsdn: Deep learning for ddos attack detection in software defined networking. 2021.

[51] Majd Latah and Levent Toker. Towards an efficient anomaly-based intrusion detection for software-defined networks. *IET Networks*, 7(6):453–459, 2018.

[52] Nisha Ahuja, Gaurav Singal, and Debajyoti Mukhopadhyay. SDN Dataset. `https://data.mendeley.com/datasets/jxpfjc64kr/1`, 2020. [Online; accessed 27-Sept-2020].

[53] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, and J. J. P. C. Rodrigues. Sdn-enabled multi-attribute-based secure communication for smart grid in iiot environment. *IEEE Transactions on Industrial Informatics*, 14(6):2629–2640, 2018.

[54] Sriram Natarajan. mgen. `https://ryu.readthedocs.io/en/latest/ryu_app_api.html`, 2014. [Online; accessed 31-Oct-2015].

[55] Zayed Al Haddad, Mostafa Hanoune, and Abdelaziz Mamouni. A collaborative network intrusion detection system (c-nids) in cloud computing. *Kohat University of Science and Technology (KUST), Proc. of International Journal of Communication Networks and Information Security*, 8(3):130, 2016.

[56] Kellie J Archer and Stanley Lemeshow. Goodness-of-fit test for a logistic regression model fitted using survey sample data. *The Stata Journal*, 6(1):97–105, 2006.

[57] Shun-ichi Amari and Si Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

[58] Songbo Tan. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28(4):667–671, 2005.

[59] Vrushali Y Kulkarni and Pradeep K Sinha. pruning of random forest classifiers: A survey and future directions. In *IEEE 2012, Proc. of International Conference on Data Science & Engineering (ICDSE)*, pages 64–68, 2012.

[60] Yuyang Zhou, Guang Cheng, Shanqing Jiang, and Mian Dai. Building an efficient intrusion detection system based on feature selection and ensemble classifier. *Computer Networks*, page 107247, 2020.

[61] Sriram Natarajan. mgen. `https://www.nrl.navy.mil/itd/ncs/products/mgen`, 2014. [Online; accessed 31-Oct-2015].

[62] Sriram Natarajan. Hping3. `http://www.hping.org/hping3.html`, 2014. [Online; accessed 21-July-2016].

[63] Hasen AlMomin and Abdullahi Abdu Ibrahim. detection of distributed denial of service attacks through a combination of machine learning algorithms over software defined network environment. In *IEEE 2020, Proc. of International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–4, 2020.

[64] Jesús Arturo Pérez-Díaz, Ismael Amezcua Valdovinos, Kim-Kwang Raymond Choo, and Dakai Zhu. A flexible sdn-based architecture for identifying and mitigating low-rate ddos attacks using machine learning. *IEEE Access*, 2020.

[65] Jin Ye, Xiangyang Cheng, Jian Zhu, Luting Feng, and Ling Song. A ddos attack detection method based on svm in software defined network. *Security and Communication Networks*, 2018, 2018.

[66] Ili Ko, Desmond Chambers, and Enda Barrett. Self-supervised network traffic management for ddos mitigation within the isp domain. *Future Generation Computer Systems*, 2020.

[67] Biao Han, Xiangrui Yang, Zhigang Sun, Jinfeng Huang, and Jinshu Su. Overwatch: A cross-plane ddos attack defense framework with collaborative intelligence in sdn. *Security and Communication Networks*, 2018, 2018.

[68] Trung V Phan, Nguyen Khac Bao, and Minho Park. a novel hybrid flow-based handler with ddos attacks in software-defined networking. In *IEEE 2016, Proc. of Intl Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 350–357, 2016.

[69] Nisha Ahuja and Gaurav Singal. ddos attack detection & prevention in sdn using openflow statistics. In *IEEE 2019, Proc. of 9th International Conference on Advanced Computing (IACC)*, pages 147–152, 2019.

[70] Nisha Ahuja, Gaurav Singal, Debajyoti Mukhopadhyay, and Ajay Nehra. Ascertain the efficient machine learning approach to detect different arp attacks. *Computers and Electrical Engineering*, 99:107757, 2022.

[71] Craig Leres. Lawrence Berkeley. arpwatch(8) - Linux man page. `https://linux.die.net/man/8/arpwatch`, 2014. [Online; accessed 31-Oct-2015].

[72] Philippe Biondi. Scapy Doc. `https://scapy.readthedocs.io/en/latest/`, 2008. [Online; accessed 29-April-2021].

[73] Prashant Kumar, Meenakshi Tripathi, Ajay Nehra, Mauro Conti, and Chhagan Lal. Safety: Early detection and mitigation of tcp syn flood utilizing entropy in sdn. *IEEE Transactions on Network and Service Management*, 15(4):1545–1559, 2018.

[74] Gaurav Singal, Vijay Laxmi, Vijay Rao, Swati Todi, and Manoj Singh Gaur. Improved multicast routing in manets using link stability and route stability. *International Journal of Communication Systems*, 30(11):e3243, 2017.

[75] Mohan Dhawan, Rishabh Poddar, Kshiteej Mahajan, and Vijay Mann. Sphinx: detecting security attacks in software-defined networks. In *Ndss*, volume 15, pages 8–11, 2015.

[76] Nisha Ahuja, Gaurav Singal, and Debajyoti Mukhopadhyay. ARP Poisoning and Flood attack in SDN. `https://data.mendeley.com/datasets/yxzh9fbvbj`, 2022. [Online; accessed 07-April-2022].

[77] Jaspreet Kaur. wired lan and wireless lan attack detection using signature based and machine learning tools. In *Springer, Proc. of Networking Communication and Data Knowledge Engineering*, pages 15–24. 2018.

[78] Anass Sebbar, Karim Zkik, Mohammed Boulmalf, and Mohamed Dafir Ech-Cherif El Kettani. New context-based node acceptance cbna framework for mitm detection in sdn architecture. *Procedia Computer Science*, 160:825–830, 2019.

[79] Han-Wei Hsiao, Cathy S Lin, and Ssu-Yang Chang. constructing an arp attack detection system with snmp traffic data mining. In *Proc. of 11th International conference on electronic commerce*, pages 341–345, 2009.

[80] Huan Ma, Hao Ding, Yang Yang, Zhenqiang Mi, James Yifei Yang, and Zenggang Xiong. Bayes-based arp attack detection algorithm for cloud centers. *Tsinghua science and technology*, 21(1):17–28, 2016.

[81] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6):352–359, 2002.

[82] Mayank Swarnkar and Neminath Hubballi. Ocpad: One class naive bayes classifier for payload based anomaly detection. *Expert Systems with Applications*, 64:330–339, 2016.

[83] Iftikhar Ahmad, Mohammad Basheri, Muhammad Javed Iqbal, and Aneel Rahim. Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection. *IEEE access*, 6:33789–33795, 2018.

[84] Zeinab Khorshidpour, Sattar Hashemi, and Ali Hamzeh. Evaluation of random forest classifier in security domain. *Applied Intelligence*, 47(2):558–569, 2017.

[85] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 2021.

[86] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.

[87] Debajyoti Mukhopadhyay, Byung-Jun Oh, Sang-Heon Shim, and Young-Chon Kim. A study on recent approaches in handling ddos attacks. *arXiv preprint arXiv:1012.2979*, 2010.

[88] Irvine. kdd dataset. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, 2020. [Online; accessed 31-Oct-2020].

[89] UNB. NSL-kdd dataset. `https://www.unb.ca/cic/datasets/nsl.html`, 2020. [Online; accessed 13-Sept-2020].

[90] Impact Trust. kyoto dataset. `https://www.impactcybertrust.org/dataset_view?idDataset=918`, 2020. [Online; accessed 31-Oct-2020].

[91] UNB. ISCX dataset. `https://www.unb.ca/cic/datasets/ids.html`, 2019. [Online; accessed 20-Aug-2019].

[92] UNB. CICIDS dataset. `https://www.unb.ca/cic/datasets/ids-2017.html`, 2020. [Online; accessed 15-July-2020].

[93] UNB. CSECICIDS dataset. `https://www.unb.ca/cic/datasets/ids-2018.html`, 2019. [Online; accessed 11-Oct-2019].

[94] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. understanding of a convolutional neural network. In *IEEE 2017, Proc. of International conference on engineering and technology (ICET)*, pages 1–6, 2017.

[95] Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404:132306, 2020.

[96] Teuvo Kohonen. Essentials of the self-organizing map. *Neural networks*, 37:52–65, 2013.

[97] Peng Li, Zhikui Chen, Laurence T Yang, Jing Gao, Qingchen Zhang, and M Jamal Deen. An improved stacked auto-encoder for network traffic flow classification. *IEEE Network*, 32(6):22–27, 2018.

[98] Jamila Manan, Atiq Ahmed, Ihsan Ullah, Leïla Merghem-Boulahia, and Dominique Gaïti. Distributed intrusion detection scheme for next generation networks. *Journal of Network and Computer Applications*, 147:102422, 2019.

[99] Ankit Agarwal, Manju Khari, and Rajiv Singh. Detection of ddos attack using deep learning model in cloud storage application. *Wireless Personal Communications*, pages 1–21, 2021.

[100] Ihsan Abdulqadder, Deqing Zou, Israa Aziz, Bin Yuan, and Weiqi Dai. Deployment of robust security scheme in sdn based 5g network over nfv enabled cloud environment. *IEEE Transactions on Emerging Topics in Computing*, 2018.

[101] Swathi Sambangi and Lakshmeeswari Gondi. A machine learning approach for ddos (distributed denial of service) attack detection using multiple linear regression. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 63, page 51, 2020.

[102] Ali Faiq Ali and Wesam S Bhaya. Software defined network (sdn) security against address resolution protocol poisoning attack. *Journal of Computational and Theoretical Nanoscience*, 16(3):956–963, 2019.

[103] Diego Kreutz, Fernando MV Ramos, and Paulo Verissimo. towards secure and dependable software-defined networks. In *Proc. of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 55–60, 2013.

[104] Alexander Shukhman, Petr Polezhaev, Yuri Ushakov, Leonid Legashev, Veniamin Tarasov, and Nadezhda Bakhareva. development of network security tools for enterprise

software-defined networks. In *Proc. of the 8th International Conference on Security of Information and Networks*, pages 224–228, 2015.

[105] Spyros Denazis, Evangelos Haleplidis, Jamal Hadi Salim, Odysseas Koufopavlou, David Meyer, and Kostas Pentikousis. Software-defined networking (sdn): Layers and architecture terminology. *Internet Requests for Comment. Jan*, 2015.

LIST OF PUBLICATIONS

[1] Ahuja, Nisha and Singal, Gaurav and Mukhopadhyay, Debajyoti and Kumar, Neeraj, "Automated DDOS attack detection in software defined networking", Journal of Network and Computer Applications (2021), Elsevier https://doi.org/10.1016/j.jnca.2021.103108, **SCI, I.F. 7.57**.

[2] Ahuja, Nisha and Singal, Gaurav and Mukhopadhyay, Debajyoti and Nehra, Ajay (2021). Ascertain the efficient Machine Learning based approach to detect different ARP attacks, Computers and Electrical Engineering, Elsevier https://doi.org/10.1016/j.compeleceng.2022.107757 (**SCIE, I.F. 3.8**).

[3] Ahuja, Nisha and Singal, Gaurav and Mukhopadhyay, Debajyoti (2021). DDOS Attack Detection using Deep Learning, Personal and Ubiquitous Computing, Springer, under review (**SCIE, I.F.3.006**).

List of International Conferences

1. Ahuja, Nisha and Singal, Gaurav (2019, December). DDoS attack detection & prevention in SDN using OpenFlow statistics. In 2019 IEEE 9^{th} Proc. of International Conference on Advanced Computing (IACC) (pp. 147-152). IEEE.

2. Ahuja, Nisha and Singal, Gaurav and Mukhopadhyay, Debajyoti. (2021, January). DLSDN: Deep learning for DDOS attack detection in software defined networking. In 2021 11^{th} International Conference on Cloud Computing, Data Science & Engineering (Confluence) (pp. 683-688). IEEE.

BIO-DATA

She pursued Bachelor in Technlogy (B.Tech) from Shri Krishna Institure of Engineering and Technology, Kurukashetra University, Kurukshetra. She pursued Master in Technology (M.Tech) from Maharishi Dayanand University, Rohtak. Her research interests include Network security, Software Defined Networking, machine learning, Deep learning, and Object Oriented Programming. She published papers in the field of Security in Software Defined Networking, Network Traffic classification Systems and Recommendation System.