Enhancing Security in Software Defined Networking using Machine Learning and Networking Algorithm

by

Mashaba Nawrin
20101088
Md. Ehtesham-ur-Rahman Aurid
20101119
Ehsan Abdullah Khan Saad
20101512
Mariea Anjuman Shrestha
20101064

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
School of Data and Sciences
BRAC University
May 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing the degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material that has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Mashaba Nawrin	Md. Ehtesham-ur-Rahman Aurid
20101088	20101119
Ehsan Abdullah Khan Saad	Mariea Anjuman Shrestha
20101512	20101064

Approval

The thesis/project titled "Enhancing Security in Software Defined Networking using Machine Learning and Networking Algorithm" submitted by

1. Mashaba Nawrin (20101088)
2. Md. Ehtesham-ur-Rahman Aurid (20101119)
3. Ehsan Abdullah Khan Saad (20101512)
4. Mariea Anjuman Shrestha (20101064)

Of Spring, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 2024.

Examining Committee:

Supervisor:
(Member)

Amitabha Chakrabarty, PhD

Professor
Computer Science Engineering
Brac University

Co-Supervisor:
(Member)

Ms. Mehnaz Seraj

S.Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

In Software Defined Networking (SDN), control planes and data planes work simultaneously; which offers flexibility and programmability, and also comes with security vulnerabilities. As big networking systems are planning to adapt SDN for its advantages, the vulnerabilities should be treated seriously. In recent years, the rapid growth of SDN has led to a USD 812.13 million market in 2022, projected to reach USD 7436.01 million by 2028. However, the susceptibility of SDN to various attacks including DDoS, Brute force attacks, link failures, Heartbleed, and session hijacking poses a significant risk. These attacks can overwhelm SDN controllers or switches, disrupting network traffic and causing service outages.
Therefore, this paper focuses on DDoS attacks on SDN. To detect and mitigate DDoS attacks in SDN, data analysis along with Machine Learning and algorithms were implemented. Here we proposed an network algorithm which works on DDoS attacks in SDN environment with the accuracy level of almost 100%.

Keywords: Software Defined Networking (SDN), Distributed Denial of Services (DDoS), Machine Learning, Network Algorithm, Decay Factor.

Acknowledgement

All glory be to the Great Allah for allowing us to conclude our thesis without any significant setbacks.

We owe a tremendous debt of appreciation to Dr. Amitabha Chakrabarty, PhD, our thesis supervisor, and Ms. Mehnaz Seraj, S.Lecturer, our thesis co-supervisor for their invaluable advice, assistance, and moral support during the duration of this study. Their professionalism, insight, and commitment to our progress have been indispensable.

We'd also like to express our heartfelt appreciation to Md. Fahim-Ul-Islam, the Research Assistant of BRAC University, for his insightful comments and ideas, which substantially enhanced the quality of our final paper.

We are grateful to our beloved BRAC University and the CSE Department for facilitating this study and giving us access to the tools we needed to complete it.

Finally, our deepest gratitude to the members of our immediate and extended families, in addition to our family members and friends, for their unending love and support, and encouragement during all of our academic endeavors.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software-defined networking (SDN) is a programmable network architecture that separates the control and data planes in the management of networks, thereby providing enhanced flexibility and programmability. There are four models of SDN. They are: **Open SDN:** Manages data plane communication between switches. Network managers use a protocol similar to OpenFlow. **SDN by APIs:** API controls data movement through devices in the network. **SDN Overlay Model:** It creates dynamic tunnels to local and remote data centers by using a virtual network on top of the existing hardware architecture. **Hybrid SDN:** Mixture of programmable networks with traditional network protocols in a single environment.

Over the past two decades, SDN has grown rapidly. The global market of SDN has grown to USD 812.13 million in 2022. Market experts predict that by the year 2028, it will reach USD 7436.01 million. However, the fundamental flexibility of Software-Defined Networking (SDN) also gives rise to potential security risks. One of the main risks of SDN is Distributed Denial of service (DDoS) attack which can damage a Software-Defined Network (SDN) by overwhelming its resources, disrupting network traffic, and causing service disruption or outages. DDoS attacks can flood an SDN controller or switch with an excessive number of requests or packets. This can consume CPU, memory, and other resources, making it difficult for the SDN infrastructure to process the right requests and maintain network configuration. The primary goal of DDoS attacks is usually to disrupt service and disable legitimate users. In an SDN environment, if an attack effectively disrupts network connectivity and overwhelms resources, service disruptions or even complete network outages can occur, affecting applications and service availability. To minimize the damage caused by DDoS attacks on SDN, network administrators must implement strong security measures, including rate limiting, traffic filtering, intrusion detection and prevention systems, and monitoring tools. Effective segmentation and isolation of network components can help control the impact of an attack.

Apart from the risk threats, SDN has a lot of advantages. As a result, many companies are switching to SDN from traditional network systems. Many companies use SDN solutions named 'Cisco open SDN Controller', 'Beacon' etc. One of the main advantages of SDN is centralized provisioning. It gives the facility to control or provision both physical and virtual elements in a single location. Moreover, in SDN, infrastructure of the network can be changed easily which increases the scalability. The optimal usage of hardware opens more fields to develop the infrastructure.

The present study analyzes the application of Software-defined Networks to enhance

the level of information security to overcome all types of attacks. This study focuses on analyzing the incorporation of SDN with Network Behaviour Testing (NBT) as a debugging tool for mitigating potential security vulnerabilities in SDN. Our work presents a thorough examination of the research environment in SDN with a particular focus on the significance of proactive failure recovery and security advancements.

Implementation of a Machine Learning model in threat mitigation would serve as a potential pattern in detecting threats in the network. Automation of the network through SDN enables it to serve as a vital function for detection and intrusion prevention. Training the models of malicious and normal network behaviors machine learning algorithms can recognize the patterns associated with potential security breaches [28]. So, implementing the Machine Learning algorithms with SDN would automate the process of detecting threats in the network. It would also enable traffic redirection and isolate compromised devices [27]. Because the SDN controller automatically adjusts security policies. As machine learning adopts new data SDN architecture becomes more resilient.So the combination of SDN and machine learning has the potential to create intelligent, robust, and secure IoT architecture.

1.1 Background

Software Defined Networking (SDN) allows centralized control and management by separating the control and data planes. Although Software-Defined Networking (SDN) presents a multitude of advantages, it concurrently raises apprehensions regarding security. Previous studies have examined several facets of Software-Defined Networking (SDN), encompassing enhancements in reliability, incident management inside industrial control systems, and the application of SDN in bolstering security for the Internet of Things (IoT). Nonetheless, it is imperative to implement proactive security measures and employ debugging tools in order to effectively tackle the ever-changing threat landscape.

1.2 Motivation

The motivation behind this study is to address the evolving landscape of networking technologies, including SDN. These technologies have become essential components of contemporary networking. Yet, they also give rise to several problems and security considerations. By exploring; the research endeavor seeks to enhance knowledge of software-defined networking (SDN) and its associated technologies, including their susceptibilities, as well as viable remedies to enhance their operational efficacy and safeguard against security threats.

1.3 Problem Statement

SDN is vulnerable to DDoS attacks because of its control plane and forwarding devices in the data plane. DDoS attack's main objective is to make the network overwhelmed with fake traffic making it inaccessible to the users to enter into the network. The main target of the DDos attack is to make SDN components such as

memory, processing, and bandwidth overflowing with traffic which degrades the network performance [28]. Through resource exhaustion, the memory gets overflowed with malicious traffic. This causes an immediate slowdown and crashes to the system. Sometimes Buffer overflow also causes the memory to not work properly. It works by sending data to the buffer then the system can handle and make the memory unusable and the system corrupted.DDoS attacks can send fragmented packets to the network to explore the network vulnerabilities in network stack implementation.

Bandwidth Exhaustion is also a major cause of DDoS attacks. In this, the network gets flooded with massive volumes of traffic. For this, it becomes difficult for the users to access the network resources.DDoS attack also tries to find vulnerabilities in the network protocols, such as SYN Flood attacks by targeting the TCP handshaking process.UDP flood attacks with stateless protocols. In the application layer, it exploits the vulnerabilities in web servers, databases, or other application components.

The aftereffects of this attack are very serious and cannot be unnoticed. The first problem would be service disruption. This exploits critical network resources and services. This can lead to financial losses, damage to reputation, and loss of customer trust. In some cases, this can lead to data breaching situations where a diversion of the mask is aimed at stealing sensitive data or compromising network security. In the aftermath of it, the operational cost becomes higher. Like restoring the services or investigating the incident the implementation of security not only becomes costly but also complex. It can also cause an organization to lose its reputation because of failure to mitigate the attack. Compliance such as GDPR, HIPAA, or PCI DSS requires implementation of proper security measures which includes DDoS attacks in the category.

So, DDOS attacks act as a vulnerability in the network and create significant challenges if not taken as a threat. If DDoS attack doesn't get mitigated the users would face service disruption, increased operational costs in an organization and regulatory compliance violations.The implementation of effective DDOS mitigation measures is important for protecting network resources and also for ensuring safer network operation.

1.4 Objective and Contributions

- **Core objective:** Comprehensive examination of security algorithms in SDN for threat handling and economic loss mitigation.
- **Contribution goal:** Enhance understanding of secured networking in different sectors and propose approaches to address security threats.
- **Specific focus:** Vulnerabilities of DDoS attack analysis in SDN using ML algorithms and handling them with network algorithms.
- **Proposed actions:** Developing an algorithm to mitigate DDoS attacks in SDN.

1.5 Algorithms, Models, and Existing Techniques:

This paper reviews several existing techniques and approaches in the field of SDN, including proactive failure recovery schemes, control plane distribution, incident response mechanisms, and comparison with similar DDoS mitigation techniques.

There is an entropy-based threshold method Feinstein and Schnackenberg (2003) [24] which motivated us to develop our custom algorithm to mitigate ddos attack. They use the threshold of network traffic to measure if the network is under attack or not, as during attack there will be major change in threshold.

Using Network Function Visualization (NFV) with SDNs improves network performance by enabling better management of network traffic flows, network visibility, and the deployment and control of network functions using software instead of hardware-specific middleboxes [19]. In this approach, to mitigate Denial of Service attacks (DoS), the SDN controllers reroute the attackers' traffic to an ICS honeypot. In the honeypot, administrators keep the attacks active to obtain information about the attacks [13].

There was also some research where we used SDN in automated Incident Response (IR) on ICS. The core concept of this approach was a playbook entry is a mapping from an incident asset type pair to a list of response actions. For each asset type response action pair, certain rules apply for the certain classes. These rules restrict the conducting of the response actions by defining respective preconditions [16].

In many SDN architectures, ML-based models were used for the security prediction. In these cases, the machine learning models were evaluated using simulated scenarios for security verification in SDN [25].

Chapter 2

Literature Review

Yuhong Li et al. discuss how incorporating a knowledge plane and IoT proxies in devices can aid in the efficient regulation of IoT applications and providers [21]. It also illustrates how SDN-based IoT supplemented by knowledge can address issues that present IoT systems encounter. The paper also discusses how the three planes can access different sorts and volumes of data in different ways for different purposes. Finally, the document analyzes how SDN may simplify and improve IoT network management.

Alireza Shirmarz et al. describe a proposed system for improving network performance improving network performance through the use of Software-defined Networking (SDN) architecture [23]. The SDN architecture consists of a controller and Forwarding Elements(FEs) that manage network traffic and routes. The suggested approach optimizes Flow Performance Requirements (FPR) and Network Status Matrics (NSM) using a mathematical model to discover appropriate path for each flow based on needed resources. With an adequate running duration, the technique improves performance parameters such as bandwidth, latency, jitter, Packet Loss Rate (PLR), utilization and BP. The document also gives a summary of relevant research as well as fundamental concepts and notations utilized in the proposed scheme.

Sen Chen et al. analyze the Dynamic Hypervisor Placement Problem (DHPP) in Software-defined Networks (SDN) and present an Integer Linear Program (ILP) to optimize hypervisor placement while taking control plane latency and hypervisor migration time into account [20]. The paper presents a through overview of related work and assesses the proposed model in real-life network topologies [20]. However, the paper may benefit from a more extensive examination of the suggested approach's limitations and potential downsides [20]. Furthermore, the study investigates different DHPP systems and provides a more in-depth analysis of the trade-offs between network performance and migration overhead [20].

Luis E. Salazar and Alvaro A. Cardenas talk about improving the security of Industrial Control Systems (ICS) that operate safety-critical infrastructures [18]. The paper discusses the need for improving the security of ICS and the importance of responding attacks [18]. It proposes a solution that involves migrating an attacker from an existing network to an ICS honeypot. The document also discusses related

work on honeypots for CPS and SDN for managing security properties of a system. It provides an overview of Software-defined Networks and the main idea behind SDN. The paper emphasizes the importance of providing seamless transition from the real system to honeypot environment, so the attacker does not detect that it has been transferred from the real target to a fake system [18].
Heng Cui et al. examine Software-defined Networking (SDN) security against fingerprinting attacks [7]. The paper describes the history of SDN and how it divides the control and data planes. The study then describes how fingerprinting attacks on SDN networks can be launched and how they can expose the network to new threats [7]. The study describes an effective countermeasure for fingerprinting in SDN networks [7]. It concludes by analyzing the consequences of fingerprinting attacks on SDN network security.

Peter Babarczi describes the architecture of durable control planes for self-driving networks [24]. The research provides a resilient hypervisor placement mechanism and looks into the challenge of determining the controller location of dynamically arriving virtual network request [24]. The paper also examines network reliability concerns and the concept of self-driving networks, which may respond to unforeseen challenges in real-time without the need for human involvement. The report finishes with simulation results and research objective for the future [24].

Pragati Shrivastava and Kotaro Kataoka propose a new "MHL fabrication" attack that can inject a bogus MHL into an SDN controller's topology view [26]. An attacker observes packets and drops LLDP messages while delivering BDDP messages. The document proposes a "Hybrid-Shield" defense mechanism that can immediately detect and block MHL fabrication attacks. The paper also goes proper potential assault scenarios as well as restrictions [26]. The move from traditional network design to SDN introduces new security challenges, and SDN architecture security is critical for effective network operation.

The Openflow architecture is originally for Local Area Networks (LANs). But it doesn't include effective mechanisms for fast resiliency. The paper implemented a link protection scheme that aims to enhance the OpenFlow Architecture by adding fast recovery mechanisms such as switch and controller [10]. The author achieved it by enabling the controller to add backup paths proactively along with the working paths and enabling the switches to perform recovery actions locally [10]. OpenFlow architecture is driven by the SDN principle of decoupling the control and data forwarding planes, OpenFlow network consists of three components **1)Openflow Switch, 2)OpenFlow Controller and OpenFlow Protocol**. 4 types of approaches were used in link protection schemes, these are **1)Restoration approach, 2)Data plane mechanism, 3)Path protection and 4)Link protection**.

Luis E. Salazar and Alvaro A. Cardenas discussed an overview and outlook of Software Defined Networking given[4], the architecture of it and the three planes; **Data, Control and Application** planes are discussed in detail. Here, OpenFlow which has a standard protocol for focused structure of SDN, control plane and open.
M. Al-Zewairi et al. proposed an experimental Software Defined Security based controller on the Open vSwitch controller to detect and prevent IP and MAC spoofing

attacks on the network [11]. The simulation result confirms that the proposed controller is capable of detecting and preventing the aforementioned attacks with high precision. The controller was implemented using Python by extending the Open vSwitch controller in Mininet. The author's proposed controller can extend any SDN controller with minimal changes to the code [11]. This controller is evaluated using a custom network topology in Mininet. In Software Defined Networks (SDN), the role of the brain of the network is the centralised controller. So, there is a risk of single-point failure.

Moazzeni et al. proposed distributed controller architecture as a possible solution to improve fault tolerance [15]. This method is called Reliable Distributed SDN (RSDN). It reliably considers data and control planes and combines controller coordination and data plane reliability to implement failure recovery actions. Usually, Industries and critical infrastructures are monitored and controlled by the Industrial Control System (ICS). There is a risk of potential cyber-attacks in these ICSs as they can cause significant physical as well as economic damage.

In the paper, A. F. Murillo Piedrahita et al. suggested an infrastructure based on SDN and Network Function Visualization (NFV) technology to design an automatic incident response mechanism that can respond to cyber attacks [13]. This paper demonstrates why SDNs and NFV together be a better solution rather than a physical hardware-specific middlebox [13]. The authors suggested mitigating the attacks using an ICS honeypot. With the help of a honeypot, administrators can keep the attack active to obtain more information about the attackers' targets and methods, while at the same time isolating and protecting the real process. There are also many threat prevention and damage controls mentioned in the paper.

The purpose of our research is to come up with weakness and opportunities to develop information security using SDN in Industrial Control Systems. In the paper [17], we got to know about the challenges of SDN such as reliability, scalability, low-level interfaces etc. As our primary focus is on ICS, we have to consider a big challenge which is an attack known as Denial of Service (DoS). SDN controls the control plane of the network centrally with a controller. When a DoS attack happens, it targets the controller causing to be overwhelmed with requests. If the controller is unable or slow to respond, severe tasks like configuration updates, routing, and traffic optimization will be difficult or impossible.SDNs rely on communication between the controller's network devices such as switches, routers, etc to secure systems and manage traffic flow. DoS attacks can disrupt this communication, preventing the controller from monitoring network resources well. This can cause network traffic to be irregular and chaotic.

According to Huseyin Polat, in the paper [22], DoS attacks can be minimized using Machine Learning. He at first created a dataset containing features and functionality of SDN normally and while under DoS attack. After training the dataset he performed machine learning models such as Support Vector Machine, Naive Bayes, Artificial Neural Network, and K-nearest Neighbors for classification models. With the highest accuracy rate of KNN, he suggested that DoS attacks can be detected easily using Machine Learning Algorithms. However, we can use our research to

build an SDN information security architecture as a service mode.

Analyzing the paper [6], we get to see authors proposing security architecture using (IAAS) to overcome security challenges such as spoofing and various types of attacks on the cloud. They do that as SPAD monitoring data traffic origin and dropping the traffic with spoofed source addresses. As mentioned above we want to build our architecture models based on SDN. SDN, unlike traditional network systems, is a programmable network system that controls hardware architecture and data traffic. The purpose of using SDN is to control hardware network architecture or creating virtual network. In the paper [5], we got knowledge of two current SDN policies which are:

OpenFlow: Decouples control and data forwarding plane. It uses flow tables with entries that specify how packets should be processed and forwarded.

ForCES: It has forwarding elements and control elements. Forces do not bring any changes in network architecture whereas OpenFlow brings.
We also learned about the usage of SDN services such as enterprise networks, data centers, SDN in facilitating network infrastructure, etc.

As SDN is a programmable software it is very possible to have bugs in the programs. If we do not be able to completely bebug our program we it will be difficult to overcome potential security challenges. To solve this problem we found the paper [3] which is about debugging SDN. Here we got to know about a debugger named NBT which is used to debug network control programmes in SDN. It uses packet backtrace to debug SDN. It analyzes the sequence of packet forwarding actions to find the problem and troubleshoot network faults. It helps SDN in troubleshooting common SDN errors, including race conditions, controller logic errors, and switch implementation errors, etc. Moreover, the combination of the Internet of Things and SNS can be used to develop unique and highly functioning helpful projects. As IoT is internet based it is not uncommon to have security issues.

This paper [19] describes potential security concerns of IoT devices and how they can be solved using SDN, As SDN gives complete network control, it is easy to address network problems. They have suggested an architecture of SDN for IoT devices that can be implemented by Rasberry Pi, Kodi Media Centre, and OpenFlow protocol.

The article titled "Modelling of OpenFlow-based Software-Defined Networks: The Multiple Node Case" is likely to focus on exploring the modeling and analysis of Software-Defined Networks (SDNs) that utilize the OpenFlow protocol [9]. It aims to provide insights into modeling and simulating SDNs, particularly in scenarios involving multiple network nodes, with the goal of enhancing our understanding of SDN behavior within complex network environments. Moreover, this article likely provides an in-depth exploration of modeling OpenFlow-based Software-Defined Networks, particularly focusing on scenarios involving multiple network nodes. It aims to contribute to the understanding of SDN behavior and performance in complex network environments.
The paper titled "Software Defined Networking Concepts" authored by Xenofon Foukas, Mahesh K. Marina, and Kimon Kontovasilis mostly centers the topic of

Software Defined Networking [8]. The management of computer networks, which involves the manual translation of high-level policies into low-level configurations, can provide significant challenges and is prone to errors. Moreover, the implementation of novel network functionalities frequently necessitates alterations to the existing infrastructure and protracted standardization procedures. Software-Defined Networking (SDN) is an approach that advocates for the implementation of programmable networks, facilitated by open network Application Programming Interfaces (APIs). This paradigm offers users a high degree of flexibility and adaptability, akin to the manner in which programming languages enable the reprogramming of computers without necessitating frequent hardware modifications. Additionally, this emphasizes the importance of utilizing open and established signaling protocols as a means of facilitating interoperability among diverse networking systems. The text examines the proceedings and emphasizes the significance of collaborative and standardized endeavors in attaining interoperability and effective communication across various networking domains. The objective is to establish a universal signaling system that can be flexibly adjusted and implemented across various networking contexts.

By implementing various models such as Decision trees, Support vector machines, and neural networks to perform classification SDN obtained network traffic data from three control systems RYU, ONU, and ODL [28]. The performance of this model was tracked. Through this procedure, the study showed improved performance from earlier studies. The study revealed notable results in Svm and Dt. But the performance wasn't very promising in the Opendaylight (ODL) controller. The study analyzes performance across various attack and target scenarios while deploying machine learning algorithms for traffic data classification in embedded system networks.

Table 2.1: Data Table of researched Papers

Ref	Task	Field	Working Method	Results
[21]	Enhancing IOT with SDN	IOT	SDN IOT based Architecture	Efficient Architecture.
[23]	Greedy flow routing algorithm in SDN	SDN	Integration IOT with SDN	Improves performance bandwidth, delay, jitter, packet loss rate (PLR), utilization, and BP.
[20]	Dynamic Hypervisor in SDN	VSDN	DHPP as an integer linear programming (ILP problem)	Balance between latency and migration cost.
[18]	Resiliency of Cyber-Physical Systems with SDN	SDN	ICS with implementation of IDS	Improves the security of Industrial Control Systems.
[7]	Fingerprinting SDN	SDN	Used on switches to delay the first few packets of each flow.	Decreases mount fingerprinting attacks in SDN networks.
[24]	Resilient Control Plane Design for VSDN	VSDN	Algorithm based on minimum set for resilient hypervisor placement.	Finds the arrival of virtual network requests and calculates the corresponding unprotected and resilient control paths.
[26]	Topology Poisoning Attacks and Prevention in SDN	SDN	Hybrid-Sheild	Utilizes existing SDN functionalities protocols to prevent MHL fabrication.
[10]	Proactive failure recovery in OpenFlow Based SDN	SDN	Enhance the openflow architecture..	Openflow controllers were extended to include backup path computation The openflow switch is extended with auto reject mechanism.

Ref	Task	Field	Working Method	Results
[4]	Composing Software Defined Networks.	SDN	Operators for modular applications in SDN	Packet model that allows programmers to extend packets with virtual fields.
[11]	Openflow based SDN, Security Challenges and Countermeasures.	SDN	Security challenges like scanning and spoofing.	Security risks in OpenFlow-based SDN.
[15]	OImprovement of SDN	SDN	Master controllers calculate the reliability of subnetworks.	To increase the reliability rate in an SDN network.
[13]	Leveraging Software-Defined Networking for Incident Response in Industrial Control Systems	ICS	Mitigate DoS attacks by rerouting the attack traffic to an ICS honeypot	Handle DoS attacks and gain information about the attacker's targets, frequency, and methods.
[17]	SDN Challenges, issues, and Solutions	SDN and hardware performance	SDN openflow policy and issues such as reliability, scalability, etc.	SDN controller ONOS is a preferable solution. Controller should be able to handle at least 100 switches.
[12]	Efficient Topology Discovery in Openflow-based SDN	OpenFlow and SDN	Efficiency in topology discovery approach in controller	Modifications reduce controller load by up to 40% .
[6]	Security as a Service Model for Cloud Environment	Information security	Current architecture of SDN networks	OpenFlow standards,SDN development tools, and languages preferable for programming SDN.
[5]	A Survey of SDN: Past, Present, and Future	SDN	Current architecture of SDN networks.	OpenFlow standards,SDN development tools, and languages preferable for programming SDN.
[3]	Debugger for SDN	SDN and Network Debugging	Debugging SDN network using nbt debugger.	Debugging tool uses packet backtrace and breakpoints to debug the network.
[22]	Detecting DDoS attacks in SDN through feature Selection and ML Model	ML and SDN	Run various machine learning models to detect attacks.	KNN classifier got the highest accuracy rate of 98.3% to detect the attack.

Ref	Task	Field	Working Method	Results
[19]	Improving Internet of Things (IoT) Security with Software-Defined Networking (SDN)	IOT	IoT devices use HTTP protocols. SDN controllers interact with the IoT devices using API and then analyze the network traffic and perform based on the configured rules.	This approach suspects Man in the Middle attacks and adds transport layer security.
[9]	Modeling of OpenFlow-based Software-Defined Networks: The Multiple Node Case	OpenFlow Protocol and SDN	Usage of OpenFlow-based SDNs, specifically in scenarios involving multiple network nodes, to enhance comprehension of SDN behavior and performance in complex network environments	The approach offers potential for network performance analysis under bursty traffic conditions, and future directions include controller modelling.
[8]	Software defined networking concepts."Software Defined Mobile Networks (SDMN) Beyond LTE Network Architecture (2015): 21-44.	SDN	Improving policy translation, seamless feature addition, protocol deployment, and adaptability to simplify network management	To solve the challenges in network design and management using SDN shifting from traditional methods.
[16]	ML based Detection and Mitigation Scheme for DoS attacks on SDN Controllers	SDN and ML	Integration of SDN and ML.It is an automated mechanism.As machine learning adopts new data SDN architecture becomes more resilient in providing proactive defense mechanisms in the Iot environment	By implementation of machine learning techniques, it enabled an automated system to detect threats and act upon the symptoms.
[25]	SDN Controllers and ML-Based Anomaly Detection in Embedded Systems: A Comparative Analysis	SDN and ML	ML-based detection and mitigation scheme for DDOS attacks on SDN controllers	Integration of SDN and ML ensures automated response mechanisms to dynamically adjust security policies.

Based on the table above we can see practical implementation of mitigating attacks in SDN is not enriched. Whereas, in traditional networks, many practical solutions are implemented. Therefore, we choose to find a solution to security threats in SDN.

Chapter 3

Methodology

Here we will present the detailed methodology of our proposed system for threat-identifying and mitigating DDoS attacks in the SDN environment. Additionally, we provide an example of datasets that we have collected and the techniques we used for the data visualization.

3.1 Dataset

Two types of datasets were used here. One of them was created by the **Communications Security Establishment (CSE)** and the **Canadian Institute for Cybersecurity (CIC) (version 2018)**. In this dataset, we found information about more than 50 devices that served as attackers and over 420 machines, and 30 servers that were victims. There were about a million data files within 6 CSV files and a total of 80 features were extracted using CICFlowMeter-V3. Another dataset name is the **"DDoS SDN dataset"**. There are 104345 rows and 23 columns. There was one noticeable column named label. Contains only 1 (malicious), and 0 (benign). With dataset packet filtering using a Machine Learning approach to classify the packets. This dataset had 3 categorical and 20 numeric features.
The datasets were created and analyzed by CICFlowMeter. This is used to generate bidirectional flows. It has information about DdoS attacks in sdn with 79 features. We have generated a heatmap for analyzing the correlation of the features in order to detect patterns of DDoS attackers and find out the dominant attributes in order to use this information for implementing security measures for SDN [14].

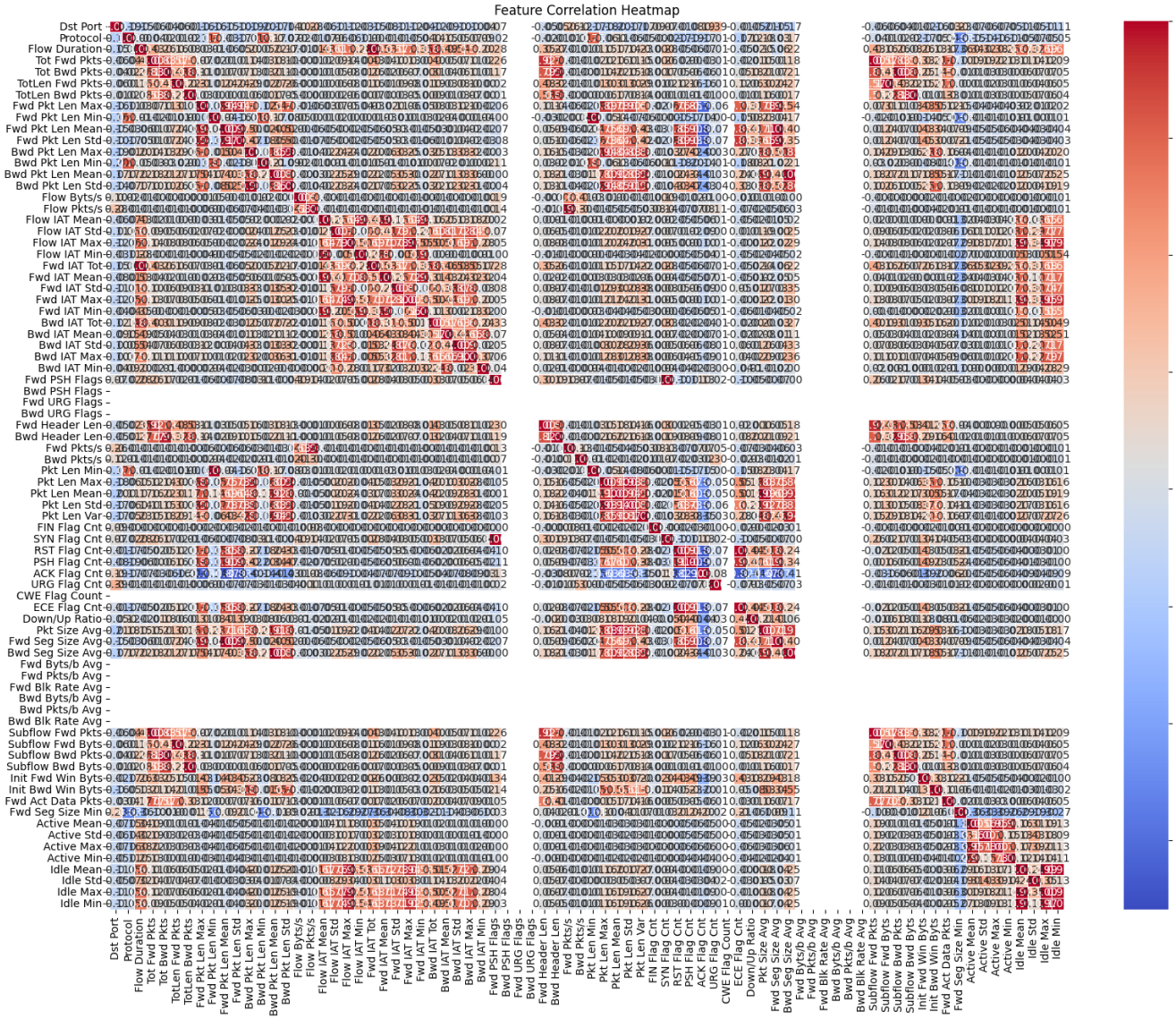


Figure 3.1: Feature Correlation Heatmap

Here in the heatmap, we used red, blue, and white to indicate the co-relations among the features. Red means positive co-relation blue means negative co-relation and white means neutral. While going through the dataset for our research; we found there are some patterns among the attackers who are doing DDoS attacks on the network. Three major features are later divided into more features so that it can be analyzed later on. Those are **Flow duration, total packets, and packet length**. In the heatmap, we see that flow duration has a weak positive correlation with total packets in a forward direction and total packets in the backward direction. This indicates that a longer flow time has more packets to send.
As flows are a bidirectional way of communication. There is a positive correlation between total number of forwarding packets and total number of backwarding packets. But during the DDoS attack time, the number of forwarding packets to the network is greater than back work packets, also the packet size is bigger than the average packet size.
The features max length of forwarding packet, min length of forwarding packets, average forwarding packet lengths, and standard size of forwarding packets have a weak positive correlation with the total size of the forwarding packets. This indicates that a larger packet size in that direction has more packets.
From our heatmap, we find out that these features share weak to moderate correlations among themselves. The features are somewhat related, we couldn't find a strong dependency on each other. However, our key observations on some strong positive and negative correlations of the features are explained below.

Strong Positive Correlation:

- Total Fwd Pkts and Fwd Pkts/s are strongly related which means forward packets increase, the forward packets per second also increase.
- Flow Duration and various IAT (Inter-Arrival Time) metrics are strongly related as IAT is derived from overall duration.
- Fwd Pkt Len Max, Min, Mean, Std have strong correlations as all these metrics describe forward packet length.
- Bwd Pkt Len Max, Min, Mean, Std; these are describing backward packet length.
- Strong correlations exist between Subflow Fwd Pkts and Subflow Bwd Pkts with corresponding total forward and backward packets.

Strong Negative Correlations:

- There is an inverse relationship between minimum forward (Fwd IAT Min) and backward inter-arrival (Bwd IAT Min) times.
- FIN Flag Cnt and SYN Flag Cnt have a negative correlation which could indicate traffic patterns for attackers.

Interesting Correlation Patterns:

- Flow IAT Std and Fwd IAT Mean have a positive correlation suggesting that the standard deviation of flow's inter-arrival times is related to the mean forward inter-arrival times.

- Subflow metrics are highly correlated with their respective total features. For example, Subflow Fwd Pkts and Total Fwd Pkts have a strong positive correlation.

Dominant attributes:

- Total Fwd Pkts and Total Bwd Pkts are highly correlated with many features like packet and flow metrics.
- Flow Duration has strong correlations with various IAT metrics.
- Fwd Pkt Len Max/Min/Mean/Std and Bwd Pkt Len Max/Min/Mean/Std have strong internal correlations within the forward and backward packet metrics.
- Subflow Fwd Pkts and Subflow Bwd Pkts correlate highly with other dominant attributes, total forward and backward packet metrics respectively.

To summarize, packet length metrics, inter-arrival time metrics, and subflow metrics are dominantly correlated within their respective groups. In order to detect and mitigate DoS or DDoS attacks in SDN using machine learning, the features will play a key role.

3.2 Dataset Pre-Processing and Visualization

In this part, we used a Machine Learning approach to analyze network traffic data, and classified packets based on protocols and sizes then we moved on to DDoS attack packets, and analyzed their size distribution to identify patterns in attack patterns. In the datasets, multiple protocols and environments were used such as HTTPS, HTTP, SMTP, POP3, IMAP, SSH, and FTP. In the majority, the traffic was HTTP and HTTPS.

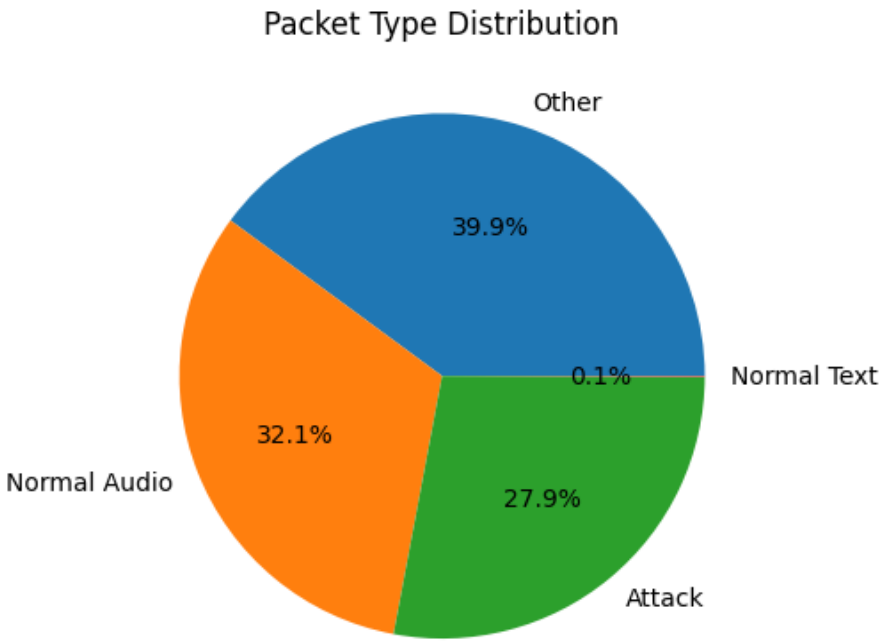


Figure 3.2: Pie-chart on packet size distribution

Here we analyzed what type of packets we used in our research. Here we could see that 0.1 percent were plain text (.txt) files, 32.1 percent were normal audio files (.mp3), 27.9 percent were attack files, and 39.9 percent were other types of files that were not identified whether they were attacking files or not.

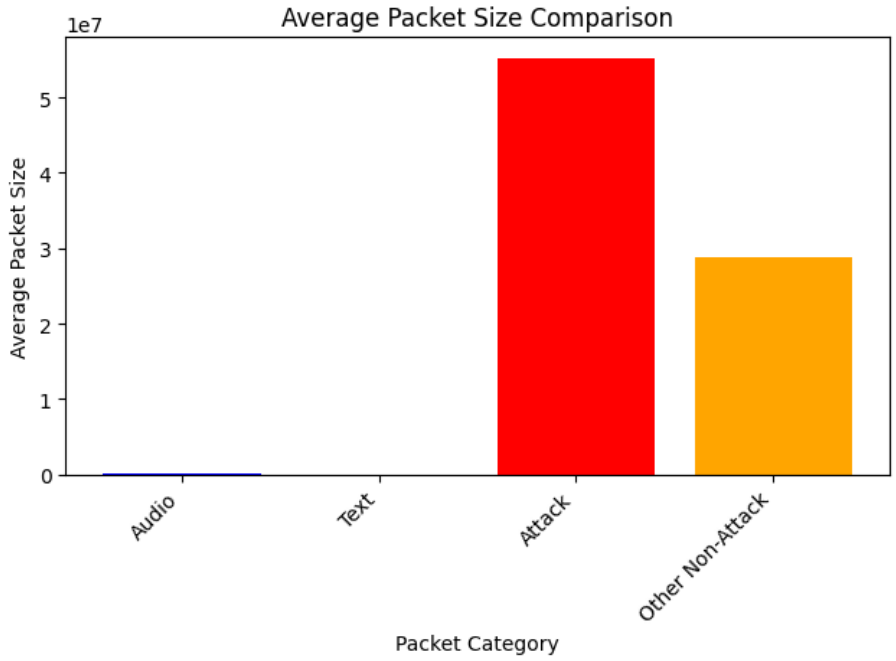


Figure 3.3: Average Packet size comparison

After analyzing the classifications of the packets, we focused on the DDoS attack packets. Here we focused on their size distribution to potentially identify packets in the attack section. Here we saw that other types of packets are most likely to be corrupted. Attackers use larger-sized packets during attacks to load the network traffic.

3.3 Packets used for DDos Detection

To detect DDOS attacks the most relevant packets to trace DDOS attacks would be HTTP Packets, TCP packets, UDP packets, and ICMP packets. A brief explanation of the packets and their behavior are given below:

1. **HTTP Packet:** HTTP-based DDoS attacks involve using tools like LOIC(Low Orbit Ion Cannon) [1]. In this process, the attacker sends a large number of HTTP requests aimed at the targeted server.
2. **TCP Packet:** TCP-based attacks occur in situations of SYN floods and are detectable by monitoring the number of TCP handshakes.
3. **UDP Packet:** UDP is flooded in the system when DDOS attacks take place. It can be detected by analyzing the UDP traffic.
4. **ICMP Packets:** ICMP flood attacks mainly take place when there is a high number of ICMP Echo Request packets overloading the bandwidth.

So, after analyzing the packets, the HTTP Packets are detected monitoring unusual spikes in HTTP request traffic [2]. TCP packets can be detected if there is the

increase in the number of SYN packets without corresponding ACK packets are a clear indicator of the attack. For the case of UDP Packets high volume of UDP Packets in random or specific ports are a clear indication of attacks through this packet [1]. At last comes the ICMP packet which can be detected by observing the number of ICMP Echo Request packets.

Chapter 4

Implementation

4.1 Work Plan for Analysis

The work plan is divided into 7 parts. Those are:

1. datasets pre-processing,
2. using the processed data in Machine Learning models,
3. extracting the features and accuracy of the Network Algorithm,
4. running the network algorithm,
5. detecting normal network packets,
6. attacking network packets,
7. mitigating them.

As all the steps are happening in the SDN environment, it is monitored centrally and keeping traces of the packets send to network. These are discussed in the next part.

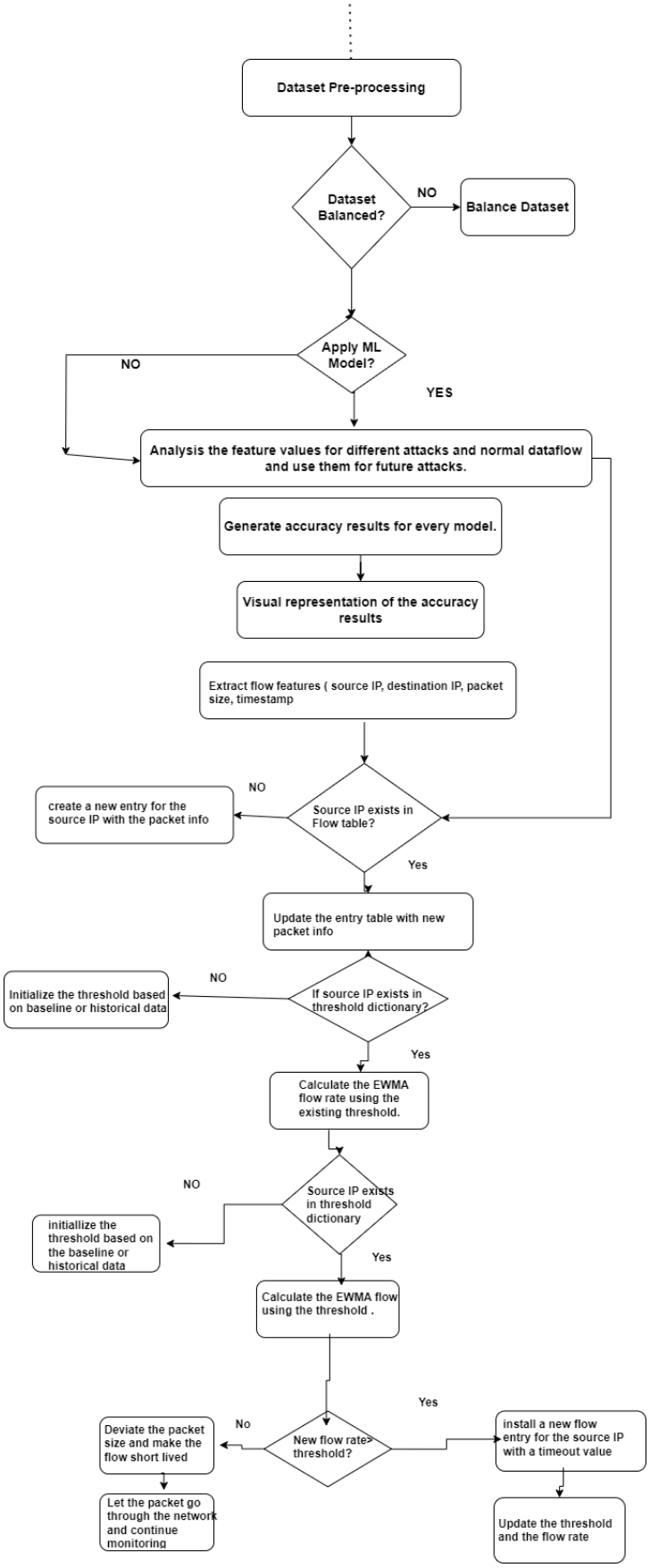


Figure 4.1: Total work flow

4.2 Strategy to Detect Attacks

In SDN, the data plane and control plane are separated to create a centralized architecture. It has a lot of advantages along with a lot of vulnerabilities. In the SDN data plane and control plane is separated and a programmed controller controls the network resources. The data plane sends raw data to the control plane[31]. Attackers try to send huge amounts of data to stop the system completely. Moreover, in order to save them from detection they send multiple attacks. As mentioned earlier our dataset contains information about seven types of attack and normal data flow. We analyzed the features of our dataset using seven Machine-Learning algorithms. The feature values varied for different attacks and normal data flow. As both forward and reverse packets were traced thoroughly along with their average, standard deviation, etc it was possible to detect attacks. After that, we made a Python script where we put our learnings from machine learning and this would create a firewall against DDoS attacks.

We have created a Python script to mitigate the attacks and continue the traffic flow in the network. Using this script, we implemented a traffic monitoring and blocking mechanism. We will discuss the scripts and algorithms in the later part.

4.3 Environment of Simulation

For our research, we used the POX controller for the mininet environment of the simulation. The reason why we chose the POX controller is because POX interacts with SDN switches using OpenFlow protocol, it has industry standards for communication between SDN controllers and switches. This helped us to simulate real-world SDN network behavior.

4.4 Application of Machine Learning

At first, we used data preprocessing which was very essential. We removed all the null values from our datasets and gave all attack packets a numerical value (the type of attacks was not specified) to analyze the impact of the attacks on the change of the values of the features. For example 0= normal data flow, 1=corrupted flow or packet. At last, we used feature scaling to bring the values of the dataset into a fixed range in order to avoid extremely large values which can corrupt the final results[29].

After that, we balanced the dataset using the Synthetic Minority Oversampling Technique for proper class distribution as the second dataset was an unbalanced one. Using smote increased the accuracy of the machine learning models we used. After proper preprocessing and balancing the dataset we used seven machine learning models, those are:

- K-Nearest Neighbours Algorithm
- Naive Bayes

- Random forest classification
- Decision tree
- Support vector machine
- Catboost
- Gradient boost.

4.5 Comparative Analysis of Machine Learning Model Used for DDoS Detection

We have trained machine learning models on the chosen datasets so that they learn the difference in the characteristics of the features in normal dataflow and attacks. The accuracy represents their detection of ddos attack accuracy.

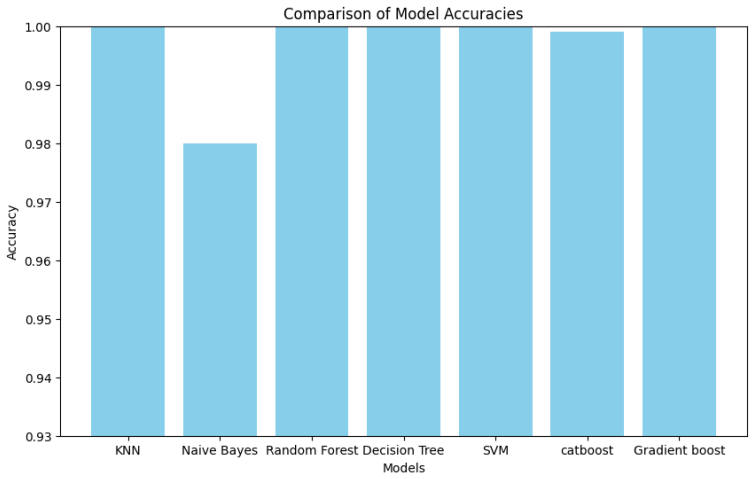


Figure 4.2: Comparison of Model Accuracies

The seven Machine Learning Models that we used each vary in their own significant value. In Decision Tree the model uses supervised learning methods for classification and regression. It is easy to interpret and visualize but may not capture complex patterns. In the graph we can see its accuracy is 100%. Now Random Forest merges multiple decision tree to get accurate and stable prediction. In the graph, the accuracy is showing 100%. SVM analyzes data for classification and regression analysis by finding the hyperplane that divides a dataset into classes. The accuracy of SVM is also 100%. KNN is a simple instance based learning algorithm that classifies a data point based on how its neighbors are classified. The accuracy is shown as 100% in the figure. Naive Bayes is a probabilistic classifier. It is based on Bayes theorem. Its accuracy is shown at about 98% in the figure. Its prediction is for high-dimensional data. However, the assumption of independence can be a limitation.

4.6 Custom Algorithm

In order to dynamically detect and mitigate DDoS attacks on SDN for real-time traffic monitoring, we have created a custom algorithm of our own. Based on our algorithm we have created a Python script that worked as a firewall to prevent the DDoS attack. To test our algorithm we used a DDoS simulation on our network which was dynamic and in our DDoS simulation we used, a random function to generate the packet flow of various sizes instead of packets with fixed same size. As our algorithm was integrated before, the DDoS attacks in the DDoS simulation were prevented. What our script does is if the packet count of an IP exceeds the threshold within the time window, it treats that as potential DDoS and blocks the IP.

Here you can see, in our SDN we have added five hosts and among five of them H1

Figure 4.3: Generating Attack hosts for a victim machine

and H2 will be sending data flood to host h3 to simulate DDoS attack and H3,H4 and H5 are normal hosts and H4,H5 will send data packet to H3 at normal rate.

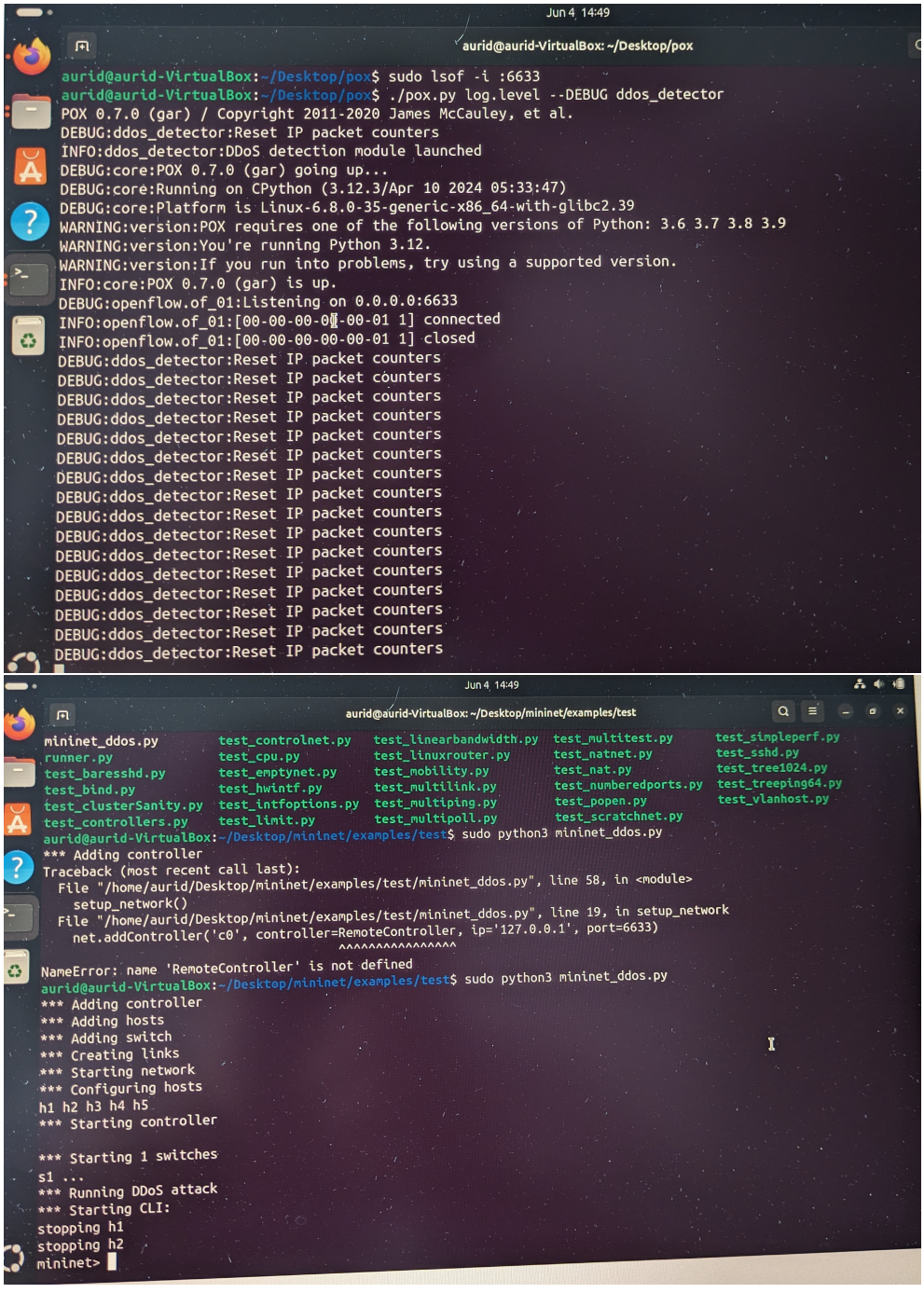


Figure 4.4: Simulation on the Device

Here our script was integrated into the POX controller and we generated data flow on the SDN which our POX controller controlled. We have trained a machine learning model random forest on our dataset to create a joblib file for DDoS prediction. We loaded that joblib file on the Python script we used on pox to detect if any host or sender is trying to do a DDoS attack on the network. And then we used its prediction in our algorithm for protecting the network. Our controller was constantly monitoring traffic flow on our sdn and it identified that h1 and h2 are sending packet flood. As a result, it stopped h1 and h2 for a time duration which we fixed in the Python script running on the controller.

4.6.1 Algorithm Description and Elements

We designed an algorithm and incorporated some techniques for more robust DDoS mitigation in an SDN environment.

1. **Data Collection and Pre-processing:** At first, we did some data collection and pre-processing. For each incoming packet to the network, we extracted some flow features such as source IP, destination IP, packet size, and timestamp. With these features, we updated the flow table entries.

2. **Anomaly detection:** Then we initialized a threshold for each source IP based on historical data or a baseline. For each flow entry, we did some calculations like EWMA **(Exponential Weighted Moving Average)** of the recent flow rate for that source IP. Every time the threshold is dynamically adjusted based on the EWMA with a decay factor (alpha, between 0 and 1). The higher alpha provides more weight to recent data.

3. **Flow characteristic analysis:** Then we analyze the flow characteristics of the packets. In this algorithm, we do not only count the packet numbers but also the standard deviation of packet sizes within the flow. Also, we monitor short-lived flows with repetitive patterns.

4. **Attack classification:** Then we combine the dynamic threshold and flow characteristic analysis to classify a flow as normal or potential DDoS.

5. **Mitigation Action:** If a flow is classified as potential DDoS, the network installs a new flow entry in the switch flow table to drop packets matching that flow's source IP, and destination IP. Then set a timeout for the flow entry to avoid permanent blocking.

6. **Adaptation and Learning:** This algorithm continuously monitors network traffic and updates historical data or baseline for dynamic threshold calculations.

Pseudocode for our custom algorithm

```
# Initialize variables
flow_table = {}  # Dictionary to store flow entries (source IP, destination IP, etc.)
thresholds = {}  # Dictionary to store dynamic thresholds per source IP
alpha = 0.1  # Decay factor for EWMA (adjust as needed)
timeout = 30  # Timeout for flow entries (seconds)

def handle_packet(packet):
    # Extract flow features
    source_ip = packet.src_ip
    # ... other flow features (destination IP, packet size, timestamp)

    # Update flow table entry
    if source_ip in flow_table:
        flow_table[source_ip].update(packet)
    else:
        flow_table[source_ip] = FlowEntry(packet)

    # Calculate EWMA flow rate for source IP
    if source_ip in thresholds:
        new_rate = calculate_ewma_rate(flow_table[source_ip], alpha)
    else:
        # Initialize threshold based on baseline or historical data
        thresholds[source_ip] = baseline_rate

    # Anomaly detection
    is_attack = new_rate > thresholds[source_ip] and \
                check_packet_size_deviation(flow_table[source_ip]) and \
                is_flow_short_lived(flow_table[source_ip])

    # Mitigation action
    if is_attack:
        install_flow_entry(source_ip, timeout)
        update_threshold(source_ip, new_rate)  # Update threshold for future packets

# Function definitions for EWMA calculation, flow characteristic checks,
# flow entry installation/removal, and threshold update (omitted for brevity)
```

Listing 4.1: Pseudocode for handling packets

4.6.2 Algorithm explanation

During the implementation phase, the POX controller was used in the Mininet simulation environment, and bespoke Python scripts were developed. The implementation of traffic monitoring and blocking methods, classification of assaults, and packet data analysis were all done using seven machine learning models. By preprocessing and visualizing network traffic data, key insights were extracted, leading to the identification of attack patterns and the selection of relevant features for analysis. During the implementation period, the POX controller was used in Mininet simulation, and Python scripts were developed for traffic monitoring, blocking, data packet analysis, and attack classification under seven ML models. A cutting-edge EWMA-based algorithm was developed to detect accuracy in DDoS attacks and boost scalability to transform attack patterns.

4.6.3 Analysis of the Algorithm

This custom algorithm tackles the critical problem of DDoS attacks in the SDN environment. DDoS attacks overwhelm a network with a flow of traffic and aim to disrupt valid connections and services. We build this algorithm upon existing approaches as we incorporate some aspects to achieve more robust DDoS detection and mitigation. In this algorithm, we used a dynamic threshold based on EWMA(Exponential Weighted Moving Average) for flow rates, like other existing network algorithms. This approach not only just counts the packets but also analyzes additional flow characteristics, leading to identifying a wider range for DDoS attacks. Using a threshold for each source IP, we improved detection accuracy.

4.6.4 Complexity analysis for DDoS mitigation Algorithm

- **Time Complexity**
  - **Dictionary update:** IP packet counts[] dictionary was used to track and constantly update the number of packets from an IP.
    - i) best case: The best case is O(1) which is the average-case complexity for dictionary operations in Python.
    - ii)worst case: O(n) in case of hash collision .
  - **Detection check:** O(1). It compares the packet count with the threshold.
  - **Blocking:** if an attack is detected, the IP address is blocked.
  - **Resetting counters:** n number of unique IP addresses tracked so O(n).
- **Space Complexity**
  - **Storage:** n number of IP addresses are stored. Hence O(n).
  - **Flow entries:** m is the number of blocked IP addresses. Hence, O(m).
- **Software complexity**
  - **Softwares used:** Pox, Mininet, virtual box.

- **Algorithm implementation complexity**
  - **Packet count:** Low. It uses a basic data structure dictionary to track packets.
  - **Mitigation:** Low, it uses flow entries to send openflow messages. When an attack is detected, it sends a flow modification message to the switch to drop the packet.
  - **Counter reset:** Low. A timer-based function resets the packet count dictionary after certain times.

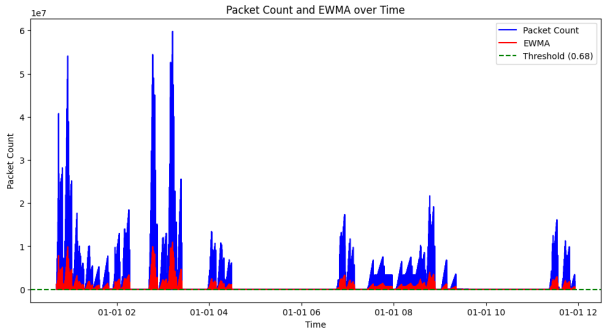
- **Hardware complexity**
  - **i)Required hardware:**
    - **Processor:** Multi-core CPU because running network simulations or using a virtual box requires moderate computational capacity.
    - **Memory:** At least 8 GB, because combined use of the virtual box, POX, and mininet requires high memory usage.
    - **Storage:** Minimum 50 GB because disk images for virtual box and log files require space.
    - **Network:** Ethernet interface for network emulation and traffic generation.
  - **Performance:** Traffic in the Mininet will mostly consume network interface resources. The physical network interface is only used if network traffic is not inside the network virtualization domain. Moreover, Disk input/output is usually low, and linked quite closely with logging and virtual operating systems.

Chapter 5

Result analysis

5.1 Result analysis

We have trained a random forest model, on our dataset in order to generate the optimal threshold for a DDoS attack. As the dataset has a targeted column, we analyzed the dataset to get what is the minimum number of packets per time window of one second for a DDoS attack which is 1148013. We used this value in our algorithm to generate a threshold limit for attack. If any IP exceeds the threshold or matches it, it blocks the IP. Below is the accuracy of the prediction of our machine learning model to detect DODS on the dataset.
Accuracy: 1.0




```
                     pktcount          EWMA
dt
1970-01-01 00:41:28   1148013  1.148013e+06
1970-01-01 00:41:29         0  9.392834e+05
1970-01-01 00:41:30         0  7.685046e+05
1970-01-01 00:41:31         0  6.287765e+05
1970-01-01 00:41:32         0  5.144535e+05
1970-01-01 00:41:33         0  4.209165e+05
1970-01-01 00:41:34         0  3.443862e+05
1970-01-01 00:41:35         0  2.817705e+05
1970-01-01 00:41:36         0  2.305395e+05
1970-01-01 00:41:37         0  1.886233e+05
                     pktcount          EWMA  is_ddos
dt
1970-01-01 00:41:28   1148013  1.148013e+06     True
1970-01-01 00:41:29         0  9.392834e+05    False
1970-01-01 00:41:30         0  7.685046e+05    False
1970-01-01 00:41:31         0  6.287765e+05    False
1970-01-01 00:41:32         0  5.144535e+05    False
1970-01-01 00:41:33         0  4.209165e+05    False
1970-01-01 00:41:34         0  3.443862e+05    False
1970-01-01 00:41:35         0  2.817705e+05    False
1970-01-01 00:41:36         0  2.305395e+05    False
1970-01-01 00:41:37         0  1.886233e+05    False
Number of packets per 1-second time window to detect potential DDoS: 1148013
```

Figure 5.1: Packet Count and EWMA over time

Here we see the visual representation of attacking packets and normal packets based on EWMA. The log report shows the packets passing through the network and checks if it is attack or not.

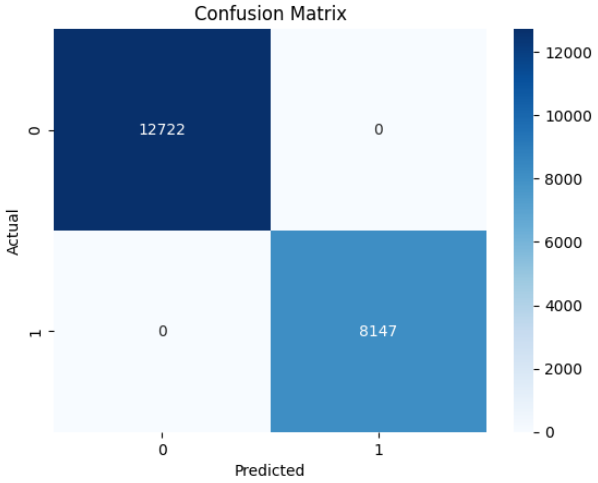


Figure 5.2: Confusion matrix to measure Detection accuracy

By analyzing our confusion matrix generated by our machine learning code we can see, that 12722 instances were correctly predicted by the model as being classified under potentially malicious traffic or DDoS attacks (True positives or TP). This implies that 8147 instances were also rightly predicted by the same model to fall under normal or benign traffic (True negatives or TN). Zero instances have been wrongly classified as positive class. In other words, the unsuspicious traffic was not identified as dangerous by our model. Zero wrongly predicted examples are negative. In other words, no actual DDoS attack was missed by the model.

5.2 Novelty analysis

DDoS attacks pose a significant threat to network infrastructure, network performance, business, etc. which can cause huge financial loss and service disruption. This algorithm is based on available solutions to DDoS attacks but it brings some unique aspects for more robust DDoS detection and mitigation. Such as:

- **Anomalies Detection Confluence:** It utilizes a fluctuating limit brought about by EWMA with regards to flow speeds, similar to some other known algorithms; this can be better understood as just a dynamic threshold. However, it transcends merely packet counts by taking into account more parameters within each flow unit such as distribution of packet sizes across different stream durations or periods. This method of examination looking at many dimensions makes recognizing multiple DDoS attack models.

- **Adaptive Threshold with Historical Learning:** It uses EWMA as a dynamic threshold that accommodates changes in network traffic. Even though this function is available on different algorithms, this one can add historical information or basic lines to develop the existing thresholds for every source IP address in particular. Consequently, the accuracy of detecting anomalies increases.

- **Flexibility and Customization:** The decay factor (alpha) for EWMA in the

algorithm can be fine-tuned to adapt it to network features and the required traffic variation sensitivity tendencies.

This algorithm gives various advantages such as flexibility, adaptability, enhanced security, etc.

5.3 Existing solutions and comparison

After much exploration of the existing solutions that are currently available here is a comparative analysis between them and our algorithm. In the next page, the comparison is shown as a tabular form.

Table 5.1: Comparison between network algorithms and proposed one

Feature/ Algorithm	Our Algorithm	NetFlow/ IPFIX	SNORT	Zeek	sFlow	DDoS Detection Hybrid Model
Type	Custom SDN-based Detection	Network Monitoring	IDS (Intrusion Detection System)	NSM (Network Security Monitor)	Network Traffic Analysis	DDoS Detection in Multicontroller SDN
Detection Method	Threshold based packet count	Flow Data Analysis	Signature-Based	Signature & Anomaly-Based	Sampling-Based Analysis	Entropy & Deep Learning (LSTM)
Scope	DDoS Attack Detection	Network Traffic Flows	Packet Payloads	Network Traffic Analysis	Network-wide Sampling	DDoS Attack Detection & Classification
Real-time Detection	Yes	Yes	Yes	Yes	Yes	Yes
Deployment Complexity	High	Medium	Medium	High	Low	Medium
Resource Usage	Moderate	High	High	High	Low	Moderate
False Positives	Low(Experimented on simple network, test needed for large case)	Moderate	High	Moderate	Moderate	Low
Flexibility	High	Moderate	Low	High	High	Moderate
Integration with SDN	Full	None	None	Partial	None	Full
Performance Impact on Network	Moderate	Low	High	High	Low	Moderate

Chapter 6

Conclusion and Future Works

6.1 Limitation

There are some limitations regarding the algorithm which depend on the chosen parameters like the decay of factor for EWMA and timeouts for flow entries. If the improper traffic arent blocked the proper way the result would show false missing DDoS attacks in the case. Constant time operation and maintaining a flow table and threshold dictionary for several active senders would create computational overhead in the system. The approach mainly depends upon predefined rules for flow characteristic analysis. Machine learning algorithms can help the system to recognize the attack pattern and adaptation over time but it requires training data and computational resources. Some advanced attackers might try to bypass the detection logic. By creating a large pool of compromised source IPs and also by changing the attacking pattern. The dataset was created by CICFlowmeter, after analyzing the dataset correlation heat map, we found that the number of strong correlations is very less compared to the number of features. This may be the reason why the prediction accuracy of our machine-learning model is high. If that is the case, then it can give false predictions when DDoS attacks with high bandwidth and large traffic occurs.

6.2 Potential for Future Improvements

In the future, there is a scoop for potential improvements in these subjects. Four key things can develop. Like, in real-time network traffic analysis, some parameters can be dynamically adjusted, for instance, the decay factor. Mechanisms for these parameters have bright sight for improvement, this is known as adaptive tuning. Secondly, to boost the accuracy of attack pattern recognition and overall detection of accuracy, machine learning integration is used, where based on the data of attacks from the past, the incorporated machine learning algorithms are trained. Thirdly, In order to overcome the high prediction accuracy, we will create our own dataset by generating and monitoring normal traffic along with DDoS simulation traffic in our own SDN environment, in the future. Then along with that and the datasets we have used, we will collect a few more public datasets that have information about DDoS attacks in SDN. Then we will implement federated learning which is a machine learning approach where all the datasets will be used to collaboratively train a single model and create the joblib file for attack prediction, without the need to centralize

the datasets. This approach will enhance the robustness and accuracy of the joblib file by leveraging diverse data sources, which will help us reduce the chance of false predictions and improve generalization across different network scenarios. We will preprocess and perform all necessary feature extraction to ensure Data Compatibility among all the datasets. Then we will use TensorFlow Federated (TFF) which is a popular framework for federated learning using TensorFlow [30]. The next idea is to discover the collaboration defensive mechanism that allows SDN controllers to communicate with detected attackers' data and strategically mitigate them. Lastly, the multi-layered defense also has future potential. For a richer defense strategy, integrate this algorithm with additional DDoS mitigation measures across multiple network levels such as resource allocation and network filtering.

6.3 Conclusion

In conclusion, this thesis has depicted a thorough methodology for monitoring and tackling DDoS assaults in Software Defined Network (SDN) environments. By adopting machine learning techniques and dynamic algorithms, the suggested system presents a robust solution to protect the network resources against malicious traffic. The research has shed light on the criticality of DDoS vulnerabilities in SDN through the diagnostics study of many datasets. Many important insights were retrieved from the preprocessing and visualization of network traffic which assisted in determining attack patterns and opting for appropriate features for research. In simple words, in our research, we used machine learning algorithms to identify the Dos attacks, and to mitigate the attacks, an attempt to introduce a sprint-based algorithm was developed. The research shows promising results, however, offering limitations and areas for further improvements, such as multi-layer defense, machine learning integration, collaboration and reputation of network systems, and adaptive tuning. The study highlights the essentiality of taking measures to prevent DDoS attacks in SDN systems. Organizations can save their network resources, maintain legal standards, and secure uninterrupted service delivery. The proposed methodology establishes an excellent foundation for future network safety research and development.

Bibliography

[1] L. Feinstein et al. "Statistical approaches to DDoS attack detection and response". In: *Proceedings DARPA Information Survivability Conference and Exposition*. Vol. 1. Washington, DC, USA, 2003, pp. 303–314. DOI: 10.1109/DISCEX.2003.1194894.

[2] Keunsoo Lee et al. "DDoS attack detection method using cluster analysis". In: *Expert Systems with Applications* 34.3 (2008), pp. 1659–1665. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2007.01.040.

[3] Nikhil Handigol et al. "Where is the debugger for my software-defined network?" In: *Proceedings of the first workshop on Hot topics in software defined networks (HotSDN '12)*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 55–60. DOI: 10.1145/2342441.2342453.

[4] Christopher Monsanto et al. "Composing software defined networks". In: *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. 2013.

[5] B. A. A. Nunes et al. "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks". In: *IEEE Communications Surveys & Tutorials* 16.3 (Third Quarter 2014), pp. 1617–1634. DOI: 10.1109/SURV.2014.012214.00180.

[6] V. Varadharajan and U. Tupakula. "Security as a Service Model for Cloud Environment". In: *IEEE Transactions on Network and Service Management* 11.1 (Mar. 2014), pp. 60–75. DOI: 10.1109/TNSM.2014.041614.120394.

[7] R. Bifulco et al. "Fingerprinting Software-Defined Networks". In: *2015 IEEE 23rd International Conference on Network Protocols (ICNP)*. San Francisco, CA, USA, 2015, pp. 453–459. DOI: 10.1109/ICNP.2015.26.

[8] Xenofon Foukas, Mahesh K. Marina, and Kimon Kontovasilis. "Software defined networking concepts". In: *Software Defined Mobile Networks (SDMN) Beyond LTE Network Architecture*. 2015, pp. 21–44.

[9] Kashif Mahmood et al. "Modelling of OpenFlow-based software-defined networks: the multiple node case". In: *IET Networks* 4.5 (2015), pp. 278–284.

[10] V. Padma and P. Yogesh. "Proactive failure recovery in OpenFlow based Software Defined Networks". In: *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*. Chennai, India, 2015, pp. 1–6. DOI: 10.1109/ICSCN.2015.7219846.

[11] Wenjuan Li, Weizhi Meng, and Lam For Kwok. "A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures". In: *Journal of Network and Computer Applications* 68 (2016), pp. 126–139.

[12] Farzaneh Pakzad et al. “Efficient topology discovery in OpenFlow-based software defined networks”. In: *Computer Communications* 77 (2016), pp. 52–61.

[13] A. F. Murillo Piedrahita et al. “Leveraging Software-Defined Networking for Incident Response in Industrial Control Systems”. In: *IEEE Software* 35.1 (Jan. 2018), pp. 44–50. DOI: 10.1109/MS.2017.4541054.

[14] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*. Portugal, Jan. 2018.

[15] D. Suleiman, Shadi Moazzeni, et al. “On reliability improvement of software-defined networks”. In: *Computer Networks* 133 (2018), pp. 195–211.

[16] Florian Patzer, Ankush Meshram, and Maximilian Heß. “Automated Incident Response for Industrial Control Systems Leveraging Software-defined Networking”. In: *ICISSP*. 2019.

[17] Deepak Singh Rana, Shiv Ashish Dhondiyal, and Sushil Kumar Chamoli. “Software defined networking (SDN) challenges, issues and solution”. In: *International journal of computer sciences and engineering* 7.1 (2019), pp. 884–889.

[18] Luis E. Salazar and Alvaro A. Cardenas. “Enhancing the Resiliency of Cyber-Physical Systems with Software-Defined Networks”. In: *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy (CPS-SPC'19)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 15–26. DOI: 10.1145/3338499.3357356.

[19] A. Al Hayajneh, M. Z. A. Bhuiyan, and I. McAndrew. “Improving Internet of Things (IoT) Security with Software-Defined Networking (SDN)”. In: *Computers* 9.1 (Feb. 2020), p. 8. DOI: 10.3390/computers9010008.

[20] S. Chen, W. Sun, and W. Hu. “On Dynamic Hypervisor Placement in Virtualized Software Defined Networks (vSDNs)”. In: *2020 22nd International Conference on Transparent Optical Networks (ICTON)*. Bari, Italy, 2020, pp. 1–5. DOI: 10.1109/ICTON51198.2020.9203137.

[21] Y. Li et al. “Enhancing the Internet of Things with Knowledge-Driven Software-Defined Networking Technology: Future Perspectives”. In: *Sensors* 20.3459 (2020). DOI: 10.3390/s20123459.

[22] Huseyin Polat, Onur Polat, and Aydin Cetin. “Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models”. In: *Sustainability* 12.3 (2020), p. 1035.

[23] Alireza Shirmarz and Ali Ghaffari. “An adaptive greedy flow routing algorithm for performance improvement in software-defined network”. In: *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields* 33.1 (2020). DOI: 10.1002/jnm.2676.

[24] P. Babarczi. “Resilient Control Plane Design for Virtual Software Defined Networks”. In: *IEEE Transactions on Network and Service Management* 18.3 (Sept. 2021), pp. 2557–2569. DOI: 10.1109/TNSM.2021.3063204.

[25] H.-M. Chuang, F. Liu, and C.-H. Tsai. “Early Detection of Abnormal Attacks in Software-Defined Networking Using Machine Learning Approaches”. In: *Symmetry* 14.6 (June 2022), p. 1178. DOI: 10.3390/sym14061178.

[26] P. Shrivastava and K. Kataoka. “Topology Poisoning Attacks and Prevention in Hybrid Software-Defined Networks”. In: *IEEE Transactions on Network and Service Management* 19.1 (Mar. 2022), pp. 510–523. DOI: 10.1109/TNSM.2021.3109099.

[27] T.G. Gebremeskel et al. “Ddos attack detection and classification using hybrid model for multicontroller SDN”. In: *Wireless Communications and Mobile Computing* (2023), pp. 1–18. DOI: 10.1155/2023/9965945.

[28] C. Gonzalez and S. M. Charfadine. “SDN Controllers and ML-Based Anomaly Detection in Embedded Systems: A Comparative Analysis”. In: *2023 10th International Conference on Wireless Networks and Mobile Communications (WINCOM)*. Istanbul, Turkiye, 2023, pp. 1–6. DOI: 10.1109/WINCOM59760.2023.10322912.

[29] T. Omar, B. Griffin, and J. Garcia. “ML based Detection and Mitigation Scheme for DoS attacks on SDN Controllers”. In: *2023 IEEE Conference on Dependable and Secure Computing (DSC)*. Tampa, FL, USA, 2023, pp. 1–6. DOI: 10.1109/DSC61021.2023.10354117.

[30] Ryhan Uddin and Sathish A. P. Kumar. “SDN-Based Federated Learning Approach for Satellite-IoT Framework to Enhance Data Security and Privacy in Space Communication”. In: *IEEE Journal of Radio Frequency Identification* 7 (May 2023), pp. 424–440. DOI: 10.1109/JRFID.2023.3279329.

[31] T. AlMasri. “IDPS-SDN-ML: An Intrusion Detection and Prevention System Using Software-Defined Networks and Machine Learning”. In: *2022 1st International Conference on Smart Technology, Applied Informatics, and Engineering (APICS)*.