Adaptive Attack Mitigation in

Software Defined Networking

Mohd Sani Bin Mat Isa

Submitted in accordance with the requirements for the degree of
Doctor of Philosophy

The University of Leeds
School of Electronic and Electrical Engineering

December 2022

The candidate confirms that the work submitted is his/her own, except where work which has formed part of jointly authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Author : Mohd Sani Mat Isa
Contributions : Proposed and implemented adaptive attack mitigation in SDN. Wrote first and final drafts of the manuscripts.

Co-author : Dr. Lotfi Mhamdi
Contributions : Provided overall supervision of the whole project. Provided feedback on technical analyses and final drafts of the manuscripts.

Co-author : Dr. Desmond McLernon

Contributions : Provided feedback on technical analyses and final drafts of the manuscripts.

Acknowledgements

My contributions, fully and explicitly indicated in the thesis, have been proposed and implemented adaptive attack mitigation in SDN. Wrote first and final drafts of the manuscripts. The other members of the group, Dr. Lotfi Mhamdi provided overall supervision of the whole project, Dr. Desmond McLernon feedback on technical analyses and final drafts of the manuscripts.

Abstract

In recent years, SDN has been widely studied and put into practice to assist in network management, especially with regards newly evolved network security challenges. SDN decouples the data and control planes, while maintaining a centralised and global view of the whole network. However, the separation of control and data planes made it vulnerable to security threats because it created new attack surfaces and potential points of failure. Traditionally, network devices such as routers and switches were designed with tightly integrated data and control planes, which meant that the device made decisions about how to forward traffic as it was being received. With the introduction of SDN, the control plane was separated from the data plane and centralized in a software-based controller. The controller is responsible for managing and configuring the network, while the data plane handles the actual forwarding of traffic. This separation of planes made it possible for network administrators to more easily manage and configure network traffic. However, it also created new potential points of attack. Attackers can target the software-based controller or the communication channels between the controller and the data plane to gain access to the network and manipulate traffic. If an attacker successfully compromises the controller, they can gain control over the entire network and cause significant disruption. Seven main categories directly related to these risks have been identified, which are unauthorized access, data leakage, data modification, compromised application, denial of services (DoS), configuration issues and system-level SDN security.

Distributed Denial of Service (DDoS) attacks are a significant threat to SDN because they can overwhelm the resources of the network, causing it to become unavailable and disrupting business operations. In an SDN architecture, the central controller is responsible for managing the flow of network traffic and directing it to the appropriate destination. However, if the network is hit with a DDoS attack, the controller can quickly become overwhelmed with traffic, making it difficult to manage the network and causing the network to become unavailable.

Coupling SDN capabilities with intelligent traffic analysis using Machine Learning and/or Deep Learning has recently attracted major research efforts especially in combatting DDoS attack in SDN. However, most efforts have only been a simple mapping of earlier solutions into the SDN environment. Focussing in DDoS attack in SDN, firstly, this thesis address the problem of SDN security based on deep learning in a purely native SDN environment, where a Deep Learning intrusion detection module is tailored to the SDN environment with the least overhead performance. In particular, propose a hybrid unsupervised machine learning approach based on auto-encoding for intrusion detection in SDNs. The experimental results show that the proposed module can achieve high accuracy with a minimum of selected flow features. The performance of the controller with the deployed model has been tested for throughput and latency. The results show a minimum overhead on the SDN controller performance, while yielding a very high detection accuracy.

Secondly, a hybrid deep autoencoder with a random forest classifier model to enhance intrusion detection performance in a native SDN environment was introduced. A deep learning architecture combining a deep autoencoder with

random forest learning feature representation of traffic flows natively was collected from the SDN environment. Publicly available packet Capture (PCAP) files of recorded traffic flows were used in the SDN network for flow feature extraction and real-time implementation. The results show very high and consistent performance metrics, with an average of a 0.9 receiver-operating characteristics area under curve (ROC AUC) recorded.

Finally, an adaptive framework for attack mitigation in Software Defined Network environments is suggested. A combined three level protection mechanism was introduced to support the functionality of the secure SDN network operations. Entropy-based filtering was used to determine the legitimacy of a connection before a deep learning hybrid machine learning module made the second layer inspection. Through extensive experimental evaluations, the proposed framework demonstrates a strong potential for intrusion detection in SDN environments.

List of Abbreviations

Acc	Accuracy
AERF	Autoencoder Random Forest
CPU	Control Processing Unit
DAERF	Deep Autoencoder Random Forest
DL	Deep Learning
DDoS	Distributed Denial of Service
DoS	Denial of Service
F1	F1-Measure
FN	False Negative
FP	False Positive
IDS	Intrusion Detection System
IOT	Internet of Things
OF	Openflow
RL	Reinforcement Learning
RNN	Recurrent Neural Network
ROC AUC	Receiver operating characteristic Area Under Curve
SDN	Software Define Network
SECOD	SDN sEcure COntrol and Data Plane
TN	True Negative
TP	True Positive

Table of Contents

List of Tables

List of Figures

Preface

To my beloved family.

CHAPTER 1 : Introduction

1.1 Motivation

Machine learning is increasingly being used to defend against Distributed Denial of Service (DDoS) attacks in Software Defined Networking (SDN) environments because it can detect and respond to attacks more quickly and effectively than traditional approaches. DDoS attacks can generate massive amounts of traffic that can quickly overwhelm network resources, making it difficult for human operators to respond in real-time. However, machine learning algorithms can analyse traffic patterns in real-time and detect anomalies that may indicate a DDoS attack. By doing so, machine learning algorithms can help detect and mitigate attacks quickly and automatically, before they can cause significant disruption to the network.

Moreover, machine learning algorithms can be trained to adapt to new types of attacks and traffic patterns over time. This means that they can become more effective at detecting and responding to attacks as they encounter more diverse and complex threats. In addition, machine learning algorithms can analyse vast amounts of data from various sources, such as network traffic logs, intrusion detection systems, and security sensors, to identify patterns and anomalies that may indicate an attack. In an SDN environment, machine learning can be used to detect DDoS attacks at various levels of the network, including the controller, the data plane devices, and the application layer. By detecting and mitigating attacks at multiple levels, machine learning can

provide a comprehensive defence against DDoS attacks, preventing them from causing significant damage to the network.

Software Define Network (SDN) has provided a massive improvement over traditional networks with its wide range of centralised controls for the monitoring and management of the entire network. Global adoption of this technology due to its unmatched benefits is expected and envisaged. To further explore the capability of machine learning to mitigate DDoS attack in SDN is the focus of this study.

1.2 Challenges

Distributed denial-of-service (DDoS) attacks are one of the most common fatal malicious attacks in the current internet environment. This is due to targeting widely distributed geographical sources and with high volume of traffic from large numbers of compromised hosts. DDoS attacks can generate massive amounts of traffic that can quickly overwhelm network resources, making it difficult for human operators to respond in real-time. However, machine learning algorithms can analyse traffic patterns in real-time and detect anomalies that may indicate a DDoS attack. By doing so, machine learning algorithms can help detect and mitigate attacks quickly and automatically, before they can cause significant disruption to the network.

Distributed Denial of Service (DDoS) attacks can have a significant impact on Software Defined Networking (SDN) environments, causing disruption to network operations, reducing network performance, and affecting the availability of critical applications and services. Furthermore, DDoS attacks can be used as a smokescreen to distract network administrators from other

types of attacks, such as malware infections, data exfiltration, and ransomware attacks. By overwhelming network resources with a DDoS attack, attackers can mask their activities and exploit vulnerabilities in the network undetected. DDoS attacks are a rapidly evolving threat, and new types of attacks and techniques are being developed by attackers all the time. As a result, it can be challenging for researchers to keep up with the latest threats and develop effective mitigation strategies. However, given the significant impact that DDoS attacks can have on SDN environments, it is crucial for researchers to continue to explore new approaches to defending against these types of attacks. This may involve developing new machine learning algorithms, leveraging advanced analytics and data visualization tools, or using innovative network security architectures to detect and mitigate attacks in real-time.

This work aims to improve and build an adaptive model which effectively mitigates DDoS attacks influenced by the above mentioned scenario and impact which focuses on combining SDN benefits and machine learning capabilities.

1.3 Objective and Scope of the Thesis

The primary objective of this research is to utilize the features offered by the SDN and harness the ability of machine learning intelligence to design and develop adaptive attack detection and mitigation for DDoS attacks. SDN centralized management and enhanced network visibility combine to develop an adaptive solution as a network application in an SDN environment. Simultaneous use in traffic monitoring and to enhance the performance of the

network by mitigating detected attacks are the true implications of the research. The adaptive approaches are hoped to improve the detection and mitigation of DDoS attacks on SDN environments. These are the predicted contributions to be made by this research:

Aim 1: Identify the attack type, detect, mitigate and evaluate its impact on the SDN environment.

i. Identify types of attack.
ii. Identify the detection mechanisms involved.
iii. Identify the mitigation approaches used.
iv. Evaluate the impact of the attack on the SDN environment by studying the bandwidth capacity and response to SLA (Service Level Agreement) by web services.
v. Summarize the impact.

Aim 2: Comparison of machine learning attack classifications in SDN environments.

i. Compare machine learning attack classifications by analysing the performance of selected algorithms.
ii. Test the classification in an SDN environment setup.
iii. Present the AUC ROC of the compared algorithms.
iv. Summarize the comparison for usage in future adaptive mitigation proposals.

Aim 3: Proposed adaptive attack mitigation in SDN

i. Introduction and research architecture.
ii. Simulation setup and performance metrics.

iii. Investigation of the effectiveness of attack mitigation in simulated environments.

iv. Summary of proposed research.

Aim 4: Experiment conducted to assess the adaptive attack mitigation in a testbed SDN environment.

i. Introduction and testbed architecture.

ii. Testbed setup and performance metrics.

iii. Investigation of the effectiveness of attack mitigation in the testbed environment.

iv. Summary of testbed deployment.

1.4 Design Challenges

Combining machine learning and SDN for DDoS attack mitigation can be a powerful approach for defending against these types of attacks. However, there are several design challenges that must be addressed to ensure the effectiveness of this approach. Some of these challenges include:

1.4.1 Statistics Collection

Design challenges: Machine learning algorithms require large amounts of data to train effectively. Collecting and analysing data in real-time from the SDN environment can be challenging, particularly when dealing with high-speed traffic.

Proposed solution: Within the capabilities of SDN, all the statistics from hosts connected to the network are available in Openflow switch, when hosts require network routing and paths for connectivity. This will provide a general

overview of all the connected hosts and the controller can make use of this information for further assembling and checking.

Benefits: With an overall view of all the connected devices and notification of traffic to the controller, the process of identifying normal and anomalous traffic can be centralized and deployed. The build-up of SDN networks with this feature can be used as the beginning of data collection for adaptive attack mitigation proposals.

1.4.2 Assessment resources

Design challenges: Traffic processing and checking will enquire a lot of resources, especially when a lot of computation and memory use is involved. This will require higher CPU and memory usage and can impact on the SDN Controller if deployed within it.

Proposed solution: A containerized application approach is proposed for best functionality. Within time intervals, the SDN Controller will call upon the adaptive module to execute the process and provide the result for further action by the controller.

Benefits: Containerized application will use minimum resources and this affects the overall setup. This approach also provides simpler deployment as the application can be downloaded and booted up as and when needed.

1.4.3 Pattern Searching

Design challenges: The general practise adopted by network administrator experts has been to manually review traffic usage and manually reassign

routes when abnormalities appear. A similar approach of focusing on prioritizing the monitoring of traffic to valid servers is the main goal. Manual methods affect the amount of time needed to analyse and proceed with further action.

Proposed solution: Automated best detection based on simplified autoencoder and entropy calculations is proposed to speed up and automate adaptive measurement using the information gathered from the training process.

Benefits: The need for the human analysis and matching findings to related abnormal traffic will be minimized. Specific pattern identified in the findings will be used as input to the adaptive method which has already been set up for particular purposes.

1.4.4 Automated Mitigation

Design challenges: Deploying network parameter tuning has always been done manually and consumes a huge amount of time and effort just to do the same test with different hosts and gaining the same action and rewards.

Proposed solution: Manual mitigation of attack will be deployed using automated containerized modules for the same purposes. This will speed up the process and ensure attack can be mitigated with less effort and fewer resources.

Benefits: Resources such as best use of time and effort can be optimized by using automated mitigation module.

1.4.5 Policy enforcement and revocation

Design challenges: : SDN environments require real-time processing of traffic to detect and mitigate DDoS attacks effectively. Machine learning algorithms can introduce processing delays, which can impact the performance and effectiveness of SDN solutions.

Proposed solution: Central views and management from a central SDN Controller would provide advantages for policy enforcement or revocation. Any new policy or revocation of policy could be done centrally from the controller. This increases the possibility of enforcing policy at the end point location of entry to the network, which is at the network port by the switch.

Benefits: Although the SDN Controller can deploy and revoke policy centrally, to any connected switch, the process of manual identifying specific hosts needs to be deployed when restricted access has been eliminated. Changes of policy enforcement to an identified host will depending on the detection and mitigation method.

1.4.6 Complete lifecycle

Design challenges: A complete approach that handles all the activities from the beginning until the end and processes them continuously in a complete lifecycle has not yet been done to provide full automation of all tasks.

Proposed solution: Machine learning attack mitigation is being proposed as automation of the complete lifecycle that handles all activities from collecting statistics, to classify, detect and mitigate threats until the necessary security policy has been implemented and taken care of.

Benefits: Any policy required to ensure an attack is mitigated, based upon findings, can be enforced immediately.

The benefits of overcoming the challenges of combining machine learning and SDN for DDoS attack mitigation can be significant. By successfully implementing this approach, advantages as below can be achieved:

i. Improve DDoS Attack Detection: By leveraging machine learning algorithms, organizations can improve their ability to detect DDoS attacks, which can reduce the impact of these attacks on their networks.

ii. Enhance Network Security: By implementing machine learning algorithms in an SDN environment, organizations can enhance their network security posture and reduce the risk of network breaches.

iii. Increase Network Resilience: By quickly detecting and mitigating DDoS attacks, organizations can increase the resilience of their networks and reduce the risk of network downtime.

iv. Reduce Operational Costs: By automating the detection and mitigation of DDoS attacks, organizations can reduce the operational costs associated with network security.

v. Improve Scalability: By leveraging SDN and machine learning technologies, organizations can improve the scalability of their network security solutions and ensure that they can handle increasing volumes of traffic.

vi. Enhance Overall Network Performance: By reducing the impact of DDoS attacks on their networks, organizations can improve overall

network performance and ensure that critical business operations are not disrupted.

1.5 Thesis Outline and Contributions

This PhD thesis describes the research carried out in the development of an adaptive attack mitigation in SDN environment. Chapter 2 discusses the SDN architecture and its security issues related to DDoS attack with further explanation on the machine learning and deep learning approach. A literature overview about machine learning adoption in a DDoS attack mitigation within SDN is also presented in same chapter. Chapter 3 looks at the potential of applying DL for intrusion detection in the a native SDN environment. The chapter further explore the improvement of detection method by using real-world data. Chapter 4 is the outcome of building up a framework for attack detection and mitigation in SDN environment with a multi-layer detection mechanism proposed. Chapter 5 concludes the thesis and gives further research directions.

The main contribution of the work is outlined below.

- An adaptive framework for the SDN environment that can collect the important network parameters and monitor the whole network for intrusion detection was put forward. The data collected can then be processed to detect abnormalities in traffic transactions which it can then respond to and mitigate promptly.
- A hybrid ML was developed with a combination of mixed approaches to assess the traffic status through flow-based anomaly detection. The

hybrid approach was shown to function with minimum impact on the overall SDN architecture.

- Recorded real-world traffic pcap with a range of simulated potential attacks was used to replicate all current potential real attack scenarios.

1.6 Publications

The work undertaken in this thesis has resulted in the following publications.

Chapter 3 : Native SDN Intrusion Detection using Machine Learning

https://ieeexplore.ieee.org/document/9306093

Published in: 2020 IEEE Eighth International Conference on Communications and Networking (ComNet)

Chapter 3 : Hybrid Deep Autoencoder with Random Forest in Native SDN intrusion detection Environment

https://ieeexplore.ieee.org/document/9838282

Published in: ICC 2022 - IEEE International Conference on Communications

Chapter 4 : Adaptive framework for attack mitigation in SDN environment

https://ieeexplore.ieee.org/document/9928595

Published in: MeditCom 2022 – IEEE International Mediterranean Conference on Communications and Networking

CHAPTER 2 : Software Defined Network

2.1 Definition

Network management involves many different tasks. The network operator configures the network devices - switches, routers, firewall and load balancer - to fulfil these tasks. The packet processing of network devices can be modelled as match-action processing, where network devices match certain patterns on packet headers. For example, the destination IP address belongs to an IP prefix and perform certain actions on the matching packets such as dropping packets or forwarding packets to an output port. Referring to the forwarding behaviour of a switch its policy will also to network-wide forwarding behaviour as the policy of the network, built upon the policies of all the switches in the network. Policies change over time because operators need to reconfigure network devices in face of various network events like traffic shifts, cyber-attacks, device failures, host mobility and so forth.

In today's network, the control plane is coupled with the data plane, as shown in Figure 2.1. The control plane on each device exchanges information with the other ones, decides how the packets should be processed on the device and configures the data plane. Since the control plane is distributed between the devices, it does not have an overall view of the network and cannot make good network-wide decisions.

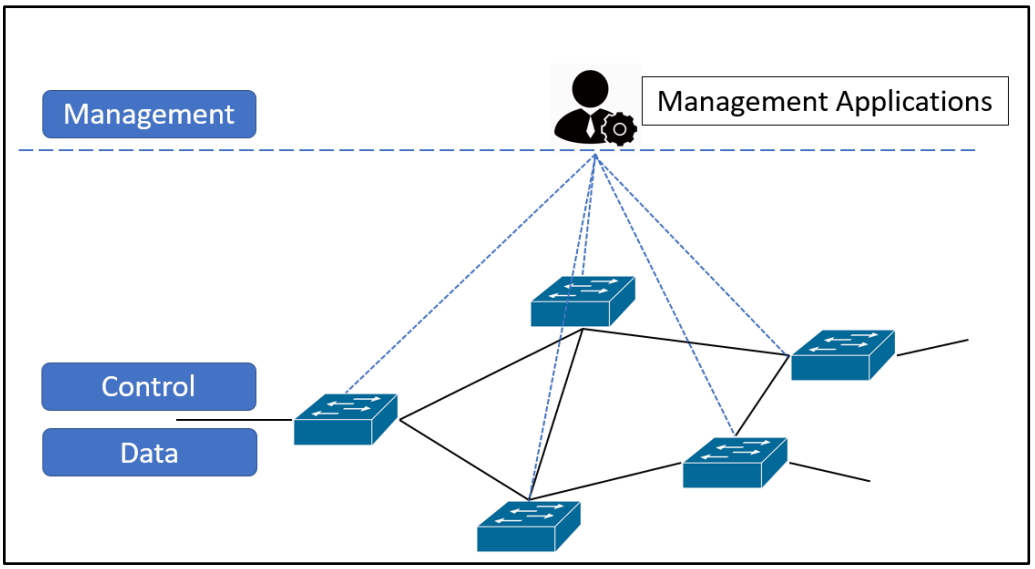


Figure 2.1 : Current Network Management Architecture

Emerging trends like network densification and service differentiation bring new challenges to future network architecture. Owing to its simplified and dynamic management, great flexibility and improved performance, SDN has attracted considerable attention in recent years. The typical three-layer model of SDN is shown in. The interface between the control Figure 2.2 plane and the data plane is open and uses open standards such as OpenFlow [1]. By separating the control plane and the data plane, SDN can offer the logical centralisation of the network management of the distributed switching devices and introduce programmability, which opens up new approaches to control functions in the application layer.

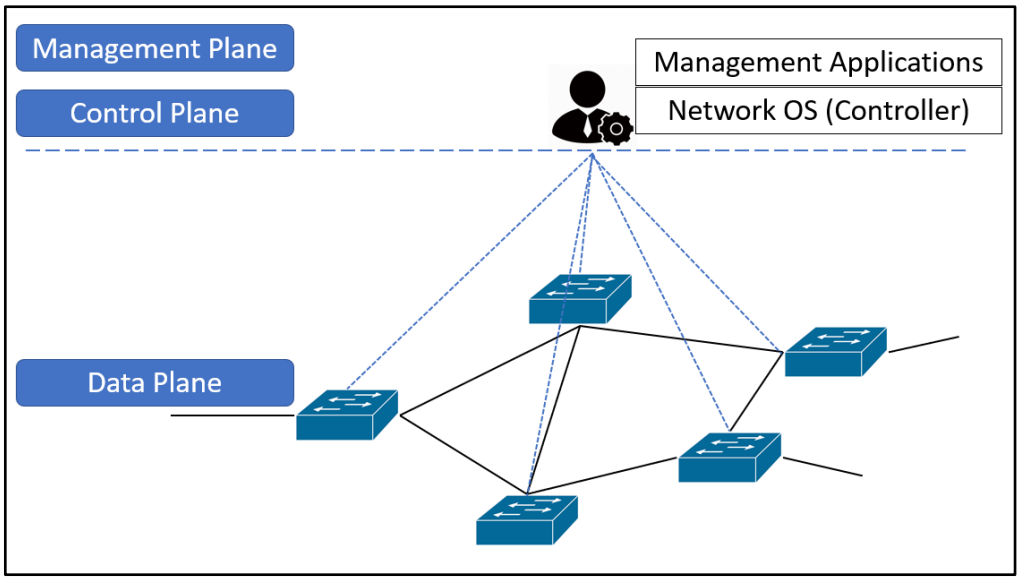


Figure 2.2 : SDN Network Management Architecture

2.1.1 Management Plane

The management plane is the set of applications that leverage the functions offered by the northbound interface to implement network control and operation logic. These include applications such as routing, firewalls, load balancers, monitoring and so forth. Essentially, the management plane defines the policies which are ultimately translated into southbound-specific instructions that program the behaviour of the forwarding devices [2].

2.1.2 Control Plane

The control plane is responsible for the network's management with the aid of the SDN controller. The SDN controller is the logically centralised intelligence within the SDN structure, so has a global view of the entire network. Thanks to its programmability, the SDN controller can also control the various functions in the application layer individually and dynamically [2]. The SDN applications are programs that directly make requests and report their

behaviour to the SDN controller. Different functions can be provided through separate program applications, such as network protocols, network monitoring and network reconfiguration.

2.1.3 Data Plane

The data plane is generally responsible for forwarding the traffic flow between switching devices based on the rules provided by the control plane. The switching devices are also responsible for collecting network state information and reporting to the control plane.

2.2 OpenFlow Protocol

The OpenFlow (OF) protocol is a standard in Software Defined Networking architecture. This protocol defines the communication between an SDN controller and the network device. The OpenFlow protocol lays down the foundation for communication between an SDN controller and a dumb network device. This protocol was first developed by researchers at Stanford University in 2008 and was first adopted by Google in their backbone network in 2011-2012. It is managed now by the Open Networking Foundation (ONF). The latest version used in the industry is V1.5.

OpenFlow is the standard southbound protocol used between the SDN controller and the switch. The SDN controller takes the information from the applications and converts them into flow entries which are fed to the switch via OF. It can also be used for monitoring switch and port statistics in network management.

The OpenFlow protocol is only operates between a controller and the switch. It does not affect the rest of the network. If a packet capture were to happen between two switches in a network, both connected to the controller via another port, the packet capture would not reveal any OF messages between the two switches. Packet capture is strictly for use between a switch and the controller. The rest of the network is not affected.

2.3 Security in SDN

A comprehensive overall high-level analysis of security in SDN is presented by the author of [3]. A total of seven threat vectors, three of which specifically refer to SDN related interfaces, were identified. Table 2.1 below, is a summary of the threat vectors.

Threat Vector	Vulnerabilities	Impact	Specific to SDN
1	Forged or faked traffic flows	DoS attack	No
2	Attacks on vulnerabilities in switches	Potentially augmented	No
3	Attacks on control plane communications	Compromised communications such as MITM attack	Yes
4	Attacks on vulnerabilities in controllers	Compromised controller jeopardized the whole network	Yes
5	Lack of mechanisms to ensure trust between the controller and management applications	Malicious applications	Yes
6	Attacks on and vulnerabilities in administrative stations	Potentially augmented	No
7	Lack of trusted resources for forensics and remediation	No assurance of fast recovery and diagnosis when faults happen	No

Table 2.1 : SDN Attack Vector

Threat is always an important aspect that needs to be handled and proper mitigation action or solutions prepared in advance and to hand. Intrusion into the network is a main concern and solutions such as putting intrusion detection mechanisms into the network is one and has become an important element in network security. Intrusion detection systems (IDS) can be in the form of dedicated devices or merely software applications that are developed for specific purposes. IDS' main function is to capture and monitor activities and events in the network and to identify possible attack that might take place. Signature-based and anomaly-based are the two types of IDS being adopted

on how to identify when the intrusion occurs. Anomaly-based IDS are mainly use in machine learning methods to classify normal and intrusion activities. Due to this approach of classifying traffic into different types, supervised learning algorithms are often used in IDS.

Machine learning is a field of computer science and artificial intelligence (AI) that involves building algorithms that can learn patterns and make predictions or decisions based on data. Essentially, machine learning involves training a computer program to learn from data, without being explicitly programmed to perform a specific task. The process of machine learning involves feeding large amounts of data into an algorithm, and then using that data to train the algorithm to recognize patterns and make predictions or decisions. The algorithm iteratively adjusts its parameters until it can accurately predict outcomes on new data. There are various types of machine learning, including supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the algorithm is trained on labelled data, meaning that the desired output is already known. In unsupervised learning, the algorithm is trained on unlabelled data and is required to find patterns and structure on its own.

Deep learning is a subset of machine learning that involves training artificial neural networks to learn from large amounts of data. These networks are designed to simulate the way the human brain works by using layers of interconnected nodes that process and transform data. The term "deep" in deep learning refers to the fact that these neural networks typically have many layers, allowing them to learn increasingly complex representations of the data as they progress through the layers. These networks are often referred to as

deep neural networks. Deep learning has had a significant impact on many fields, particularly computer vision and natural language processing, and has achieved state-of-the-art performance on many tasks, such as image and speech recognition, language translation, and game playing. Some notable deep learning architectures include Convolutional Neural Networks (CNNs) for image processing and Recurrent Neural Networks (RNNs) for sequence processing. One of the key advantages of deep learning is that it can automatically learn features from raw data, eliminating the need for manual feature engineering. This has allowed deep learning to achieve remarkable results on tasks that were previously considered challenging or impossible.

Machine learning and deep learning can also be applied to SDN to detect and mitigate Distributed Denial of Service (DDoS) attacks. DDoS attacks are a type of cyber-attack where a large number of compromised devices flood a network with traffic, overwhelming its resources and causing it to become unavailable.

Machine learning and deep learning can be applied to SDN for DDoS attack detection and mitigation in the cases as below:

1. Traffic classification: Machine learning algorithms can be trained to classify network traffic based on the characteristics of the packets, such as source and destination IP addresses, protocol type, and port numbers. This can help identify abnormal traffic patterns that may indicate a DDoS attack.
2. Anomaly detection: Deep learning models can be trained to detect anomalies in network traffic that may be indicative of a DDoS attack. These models can learn to identify patterns in traffic flows that are

different from normal traffic, such as an unusually high number of requests from a single IP address.

3. Real-time response: Once a DDoS attack is detected, SDN can use machine learning and deep learning to take real-time action to mitigate the attack. For example, the SDN controller can dynamically reroute traffic to avoid congested links, or can throttle or block traffic from suspicious IP addresses.
4. Proactive mitigation: Machine learning and deep learning can be used to analyse historical data and predict potential DDoS attacks before they happen. This can allow SDN to proactively allocate resources and configure network policies to prevent or mitigate future attacks.

Machine-based IDS learning in SDN has been carried out by many researchers. The focus of their studies has been divided into three objectives - classifying traffic as normal or anomalous, classifying intrusions as normal or anomalous, grouping the details into attack types, but focusing only on the detection of DDoS attacks. Work in [4] propose a Hidden Markov Model (HMM) for classifying malicious activities in the form of a network intrusion detection system (NIDS). The HMM-based model used five selected flow features to determine the status of the packet being analysed. Each of the selected feature is treated as independent event. Potential intrusion connections and vulnerable hosts are predicted by the author in [5] using 4 machine-learning algorithms. The Decision Tree algorithm, Bayes Net algorithm, Decision Table algorithm and Naive Bayes algorithm were compared. The results from the algorithms are then used by the SDN

controller to define the policy by blocking an entire subnet from accessing the vulnerable hosts identified. This strict action of blocking an entire subnet also affects other, valid users in the subnet, so should be fine-tuned for more specific action. Deep Neural Network adoption has also been experimented with by the researchers in [6] and [7]. Both used 6 basic flow features to prove that good performance anomaly detection can be achieved, showing that such an approach has a good chance of being deployed in future, but more experiments with real traffic in line-rate operation to compare to the NSL-KDD dataset being used need to be explored.

Identifying different types of attack after distinguishing between normal and intrusion traffic was studied in [8]. The author proposed an improved behaviour-based SVM for categorizing network attacks. A decision tree is used to select the most relevant features before becoming the training input for the proposed work. Promising results were reported but comparisons made between SVM and others ML algorithms have done been detailed out. Another method known as Non-symmetric Deep Auto-Encoder (NDAE) was proposed by [9], combining deep learning and random forest learning to speed up detection while still maintaining high accuracy. Both KDD-99 and NSL-KDD datasets were used for research. The results were compared with several other previous research but the performance of the proposed NDAE has not been verified within a real-world network environment.

Research related to DDoS attack detection also has been carried out by various researchers. As mentioned in the example in the introduction above, DDoS attacks are a huge problem and highly threaten the network. A lightweight DDoS attack detection was proposed in [10] using a NOX

controller in SDN. Traffic flow features were collected from OpenFlow switches and an unsupervised ANN introduced using the Self Organizing Maps (SOM) method for attack detection. A deep learning model incorporating RNN and CNN was used by [11] to detect DDoS attacks in SDN. The model consists of an input layer, a forward recursive layer, a reverse recursive layer, a fully connected hidden layer and finally, an output layer. Feature reductions are used in the model to detect attacks. Another SDN-based deep NN research was carried out by [12] proposing 8 classes of output layer, one of normal traffic and 7 types of DDoS attack. A total of 68 flow features were collected from network traffic and several algorithms compared. With the number of selected features, additional overheads for the controller are likely and the extra burden will require more processing time. Collaborative DDoS mitigation mechanisms in SDN using machine learning was elaborated by [13] with the support of multiple SDN controllers, namely Ryu, Pox, ONOS and OpenDayLight in separate locations. An authorization module was also introduced to verify communication between the controller and managed OpenFlow switches. A total of 25 selected features from the NSL-KDD dataset were chosen for validation. With a traffic size of 3000 Mbps, the usage of CPU in the model reached up to 90% as the consequence of handling a large number of features to computerize.

The related research for security in SDN by adapting the machine learning approach is shown in Table 2.2, below.

Ref.	Dataset	Input Features	Output Features	Algorithm	Findings
[13]	Self-collected	5	2 (normal, anomaly)	HMM	Each feature is independent
[14]	Longtail Log Analysis	4	2 (normal, anomaly)	C4.5, DT, BN, NB	Strict blocking affects entire subnet
[15]	NSL-KDD	6	2 (normal, anomaly)	DNN	Highly complex with few features selected
[16]	NSL-KDD	6	2 (normal, anomaly)	RNN	Highly complex with few features selected
[17]	KDD-99	23	5 (normal, 4 attack types)	SVM	No comparison with other ML
[18]	KDD-99, NSL-KDD	41	5 (normal, 4 attack types)	RF	Not verified within real-world environment
[19]	Self-collected	6	2 (normal, DDoS)	SOM	No attack source detection
[20]	ISCX Dataset	20	2 (normal, DDoS)	DNN	Highly complex but high number of features selected
[21]	Self-collected	68	8 (normal, 7 attack types)	DNN	High overheads with high number of features selected
[22]	NSL-KDD	18	2 (normal, anomaly)	NB	High overheads with 90% CPU

Table 2.2 : Machine Learning Approach to SDN Security

2.4 Attack types, detection, mitigation and evaluate on SDN environment

2.4.1 Definition

A Distributed Denial of Service (DDoS) attack on Software-Defined Networking (SDN) is a type of cyber-attack that aims to disrupt the operation of an SDN network by overwhelming it with a high volume of traffic from multiple sources. In an SDN architecture, the network control and data planes are separated, and the control plane is managed by a centralized controller that communicates with the data plane switches to forward traffic.

In a DDoS attack on SDN, the attacker may target the SDN controller, switches, or both, by flooding them with a large volume of traffic that exceeds their capacity to handle it. This can cause the controller or switches to become unresponsive, resulting in a denial of service for legitimate users.

The attack can be carried out using various techniques, including flood-based attacks, amplification attacks, TCP SYN flood attacks, and resource depletion attacks. These attacks can exploit vulnerabilities in the SDN architecture or target specific resources, such as the OpenFlow protocol or the SDN switches.

To mitigate the risk of DDoS attacks on SDN, network administrators can implement various security measures, such as deploying firewalls, intrusion detection and prevention systems, and traffic filtering mechanisms. They can also adopt a multi-layered defence approach and implement network segmentation to isolate critical assets from potential attack vectors. Increasing reliance on the Internet, especially social media, with a massive increase in

Internet of Things (IoT) devices accessing widely distributed data centres has aggravated this problem. For example, In 2016, the Mirai botnet launched a series of DDoS attacks on several targets, including the DNS service provider Dyn. The attack exploited vulnerabilities in IoT devices to form a botnet that flooded Dyn's servers with traffic, causing several major websites, such as Twitter and Reddit, to become unavailable for several hours. The attack also targeted the SDN controllers and switches of the affected networks[14]. Another example is the DDoS attack on February 28, 2018, when the attackers exploited the vulnerability of memory crashed, and deployed a heightened attack from UDP port 11211. The attack targeted GitHub's SDN infrastructure and flooded its servers with traffic from compromised botnets. As reported by Verisign [16], Q2 2018 DDoS Trends show an increase compared to Q1 2018, with the largest number of attacks dominated by the UDP flood attack. As shown in Figure 2.3, the ever-increasing volume of attacks recorded is a clear sign that the problem exists long-term, and solutions need to be put in place to handle or mitigate it. These real-world cases demonstrate the potential impact of DDoS attacks on SDN networks and the importance of implementing appropriate security measures to mitigate the risk.

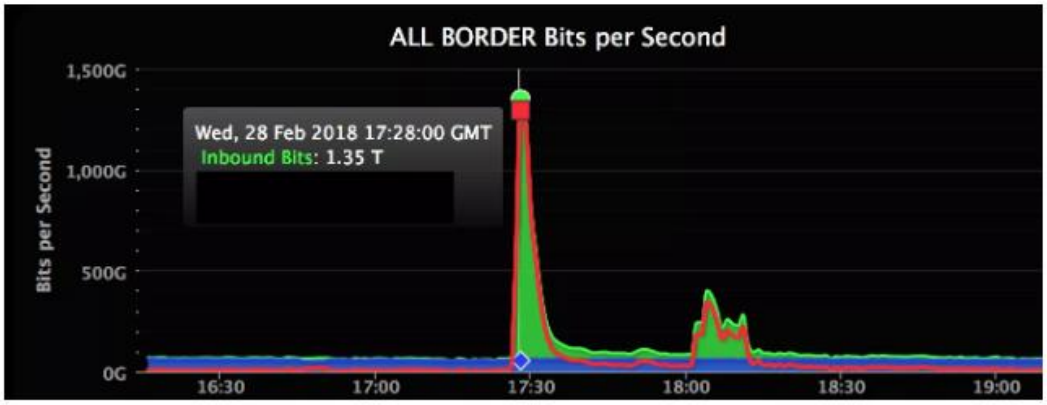


Figure 2.3 : DDoS attack on Github

2.4.2 Attack Method

An attack on SDN is a deliberate attempt to compromise or disrupt the operation of an SDN-enabled network. There are various types of attacks that can be launched against an SDN, including:

i. DoS or DDoS

The most common and devastating types of attacks that can be launched against a network, including SDN. These attacks aim to disrupt network operations by overwhelming the network with traffic or requests, rendering it unable to respond to legitimate traffic. In a DoS attack, a single attacker floods the network with traffic or requests from a single device, such as a botnet or a zombie computer. This can consume all available network resources, such as bandwidth, processing power, or memory, and cause the network to become unresponsive or crash. In a DDoS attack, multiple devices are compromised and coordinated to simultaneously flood the network with traffic or requests. This type of attack is more difficult to detect and

mitigate because the traffic originates from multiple sources, making it difficult to block or filter.

There are several types of DDoS attacks, including:

- Volume-based attacks: These attacks aim to overwhelm the network with a large volume of traffic, such as UDP floods, ICMP floods, or SYN floods.
- Protocol-based attacks: These attacks exploit weaknesses in network protocols to consume network resources, such as Ping of Death, Smurf attacks, or DNS amplification attacks.
- Application-layer attacks: These attacks target specific applications or services, such as HTTP floods, Slowloris attacks, or RUDY attacks.

To defend against DoS and DDoS attacks on SDN, several techniques can be used, such as rate limiting, traffic filtering, and load balancing. SDN can also be used to dynamically reroute traffic and allocate resources to mitigate the impact of these attacks in rea-time.

ii. Man-in-the-middle (MitM) attacks

A type of cyber-attack in which an attacker intercepts and modifies network traffic between two hosts, allowing the attacker to eavesdrop on or alter the communication. MitM attacks can occur on any type of network, including SDN, and are particularly dangerous because the attacker can potentially gain access to sensitive information such as

login credentials, financial information, or other personal data. In a typical MitM attack, the attacker positions themselves between two communicating hosts, such as a client and a server. The attacker intercepts the traffic passing between the two hosts and may modify the traffic to suit their objectives. For example, the attacker may capture sensitive information such as login credentials, credit card numbers, or other personal data. They may also alter the communication to carry out other attacks, such as injecting malware or performing a phishing attack. MitM attacks can be carried out using various techniques, such as:

- ARP spoofing: In this technique, the attacker sends fake Address Resolution Protocol (ARP) messages to the hosts in the network, tricking them into associating the attacker's MAC address with the IP address of the legitimate device. This allows the attacker to intercept and modify network traffic.
- DNS spoofing: In this technique, the attacker alters the DNS resolution process so that the victim is directed to a fake website that the attacker controls. The attacker can then intercept and modify the communication between the victim and the fake website.
- SSL stripping: In this technique, the attacker intercepts the SSL communication between the two hosts and downgrades the communication to an unencrypted form. This allows the attacker to eavesdrop on the communication and potentially capture sensitive information.

To prevent MitM attacks, it is important to use encryption and authentication techniques such as SSL/TLS, SSH, and IPsec. In SDN, these techniques can be implemented using various security protocols such as OpenFlow Secure Channel (OF-SC) and OpenFlow Management and Configuration Protocol (OF-MCP).

iii. Spoofing attacks

A type of cyber-attack in which an attacker impersonates a legitimate user or device on the network in order to gain unauthorized access or execute other attacks. Particularly dangerous because they allow the attacker to bypass security controls and gain access to sensitive information or systems. The attacker uses various techniques to impersonate a legitimate user or device on the network. For example, the attacker may:

- IP spoofing: In this technique, the attacker modifies the source IP address of their network packets to make them appear as if they originated from a trusted source on the network. This allows the attacker to bypass network security controls, such as firewalls and access control lists, and gain unauthorized access to network resources.
- MAC spoofing: In this technique, the attacker modifies the MAC address of their network interface to make it appear as if it belongs to a trusted device on the network. This allows the attacker to bypass security controls that use MAC addresses for authentication, such as port security and MAC filtering.
- DNS spoofing: In this technique, the attacker alters the DNS resolution process so that the victim is directed to a fake website

that the attacker controls. The attacker can then impersonate the legitimate website and capture sensitive information such as login credentials or financial information.

iv. Replay attacks

A type of cyber-attack in which an attacker captures and replays network traffic to gain unauthorized access or execute other attacks. Replay attacks can also occur on SDN, and are particularly dangerous because they allow the attacker to bypass security controls and gain access to sensitive information or systems.

The attacker captures a network packet that contains sensitive information, such as login credentials or financial information. The attacker then replays the captured packet at a later time to gain unauthorized access or execute other attacks. For example, the attacker may replay a captured packet that contains login credentials to gain access to a network resource.

Machine learning algorithms can be trained to recognize patterns in network traffic that indicate a replay attack, such as the same packet being sent multiple times. When an abnormal behaviour is detected, the system can take immediate action, such as blocking the traffic or alerting the network administrator. By using machine learning and deep learning techniques, SDN networks can improve their ability to detect and mitigate replay attacks, thereby enhancing their overall security posture.

v. Malware attack

An attacker infects network devices with malware, allowing them to gain control of the device or use it to launch further attacks. Malware is a type of software that is specifically designed to damage, disrupt, or gain unauthorized access to a computer system or network.

The attacker uses various techniques to infect network devices with malware. For example, the attacker may send a phishing email or a malicious attachment that, when opened, installs malware on the victim's device. The attacker may also exploit a vulnerability in the network device's software to gain unauthorized access and install the malware. Once the malware is installed on the network device, the attacker can use it to gain control of the device or use it to launch further attacks. For example, the attacker may use the infected device to launch a DDoS attack, steal sensitive data, or install additional malware on other network devices.

vi. Configuration attacks

Attacker modifies the configuration of network devices or the SDN controller to disrupt or compromise network operations. In SDN, network devices are configured and managed by the SDN controller, which provides a central point of control for the network. The attacker gains unauthorized access to the SDN controller or a network device and modifies its configuration settings. For example, the attacker may change the routing configuration to redirect traffic to a malicious destination, or modify access control settings to allow unauthorized

access to network resources. These types of attacks can have serious consequences, as they can disrupt network operations and compromise the confidentiality, integrity, and availability of network resources. In addition, they can be difficult to detect and mitigate, as they often involve subtle changes to network configurations that can go unnoticed for long periods of time. To prevent configuration attacks, it is important to implement strong access control and authentication mechanisms to protect the SDN controller and network devices from unauthorized access. This can include using strong passwords, two-factor authentication, and other security measures to prevent unauthorized access.

2.4.3 Attack Detection Method

SDN has many distinctive features which are key to detecting and mitigating attacks, including separation of the control plane from the data plane from the logically centralized controller. Thus, allowing the programmability of the network by external applications, using software-based traffic analysis and the capability to dynamically update forwarding rules. Attack detection method are elaborated below.

i. Entropy

Entropy-based methods depend on network feature distribution to detect any anomalous network activities. The probability distribution of various network features such as the source IP address, destination IP address, and port numbers are used to calculate the

entropy. Predefined thresholds of changes in entropy values are used to identify the presence of anomalies.

ii. Machine Learning

Machine learning-based methods employ techniques such as Bayesian networks, SOM and fuzzy logic to identify the presence of anomalies. These algorithms consider various network features and traffic characteristics to detect the presence of anomalies.

iii. Traffic pattern analysis

These techniques work on the assumption that infected hosts all exhibit similar behavioural patterns which are different from those of benign hosts. Typically, in the case of a botnet attack, infected machines are usually controlled by a single bot master. Similar traffic patterns are observed because of the command is sent to many members of same botnet, so causing the similar behaviour.

iv. Connection rate

These techniques are classified into the connection success ratio and the connection rate. The connection rate refers to the number of connections set up within a certain window of time.

v. Integrated Snort and OpenFlow

This technique uses a combination of IDS and OpenFlow to detect attacks and reconfigure the network dynamically. An IDS monitors the traffic to identify malicious activities. OpenFlow switches are

then dynamically reconfigured based on the detected attacks in real time.

2.4.4 Attack Mitigation Approach

SDN has been focused on to improve the agility and flexibility of a network. It empowers networks to respond quickly to changing network requirements via a centralized controller. The SDN controller provides a global view of the network. Further, the notion of the centralized controller leads to consistent configuration throughout the network, since all network policies are defined by a centralized controller, it not only simplifies anomaly detection, but also facilitates the prompt invocation of mitigation mechanisms. For example, when a DDoS attack is detected, a threat mitigation application may effectively reprogram switches to block malicious traffic. Types of attack mitigation actions are listed below.

i. Drop packets

   The network traffic conforming to the defined rules is transmitted while any remaining is dropped.

ii. Block port

   The network traffic from attacking port is completely blocked.

iii. Redirection

   The legitimate traffic is redirected to a new IP address.

iv. Control Bandwidth

The controller limits the flow transmission rate by allocating the average bandwidth to each interface.

v. Network reconfiguration and topology change

The network controller changes the flow table of each switch to change the network topology.

vi. Deep packet inspection

Deep packet inspection is a process that can examine both the header and data part of a packet. Deep packet inspection enables security to function and makes it possible to detect several types of attack, including buffer overflow attacks, denial-of-service attacks and worms and virus attacks.

vii. MAC address change and/or IP address change

When an attack is detected, the MAC address or IP address of the victim is changed. Legitimate traffic is routed to a new address and malicious traffic it blocked.

viii. Quarantine or Traffic isolation

This mitigation technique prevents the network resources from being overwhelmed by a volume-based attack by isolating the malicious traffic.

Machine Learning (ML) in SDN refers to the application of machine learning techniques and algorithms to optimize and enhance various aspects of SDN operations and management. By leveraging ML, SDN environments can dynamically adapt and improve their performance, resource allocation, security, and decision-making processes. A few key areas where machine learning is commonly applied in SDN:

i. Traffic Engineering: ML algorithms can analyse network traffic patterns, predict future traffic demands, and optimize the routing and resource allocation accordingly. This enables efficient utilization of network resources, load balancing, and improved Quality of Service (QoS).

ii. Network Security: ML techniques can be used to detect and mitigate network security threats in real-time. By analysing network traffic and behaviour patterns, ML algorithms can identify anomalies, detect potential DDoS attacks, intrusion attempts, or abnormal activities, and trigger appropriate security measures.

iii. Network Performance Optimization: ML algorithms can analyse network performance metrics, such as latency, packet loss, or throughput, and identify patterns or correlations that impact network performance. Based on these insights, ML can dynamically adjust network configurations, routing decisions, or resource allocation to optimize performance.

iv. Fault Detection and Management: ML can help in identifying network faults or failures by analysing network data and performance metrics. ML algorithms can learn normal network behaviour and detect deviations, enabling proactive fault detection, root cause analysis, and efficient network troubleshooting.

v. Network Resource Management: ML techniques can analyse historical network usage data and predict resource demands. This allows for efficient resource provisioning, capacity planning, and dynamic scaling of network resources based on traffic patterns and predicted future demands.

vi. Policy and Intent-Based Networking: ML can be utilized to translate high-level policies or intents into concrete network configurations. By learning from historical data and network behaviour, ML algorithms can automate the translation and enforcement of policies, simplifying network management and reducing manual configuration efforts.

2.5 Related Research

The drawbacks of both anomaly-based and signature-based detection methods have been elaborated by [17] . Anomaly detection has a high false alarm rate because it may categorize activities which users rarely perform as anomalous. On the other hand, signature original multi-agent router throttling based on the divide-and-conquer paradigm to eliminate detection risks cannot discover new types of attack as it uses a database of patterns of well-known attacks. Therefore, the research proposed an IDS that can identify known and unknown attacks effectively by combining features of both anomalous and signature detection using log files. The proposed IDS is based on collaboration between the RL method in association with rule learning and log correlation techniques. Positive or negative rewards are granted by the RL when the algorithm selects log files that contain anomalies or any signs of attack or not, respectively accurately. This procedure enables learning by

experience of the system as it learns to choose more appropriate log files when searching for traces of attack.

As mentioned in the introduction, DDoS is one of the most destructive attacks in the current internet environment. This is due to dealing with widely distributed geographical sources with high volumes of traffic coming from large numbers of compromised hosts. A new machine learning-based collaborative DDoS mitigation mechanism in SDN was propose in [13]. A model for DDoS detection in SDN was created using an NSL- KDD dataset and after training the model on this dataset, real DDoS attacks were used to assess the viability of the proposed model. The results show that the proposed technique compares favourably with current techniques, having better performance and accuracy.

This thesis aims to improve on and build an adaptive model which can effectively mitigate attacks, influenced by the above mentioned research, focusing on combining the benefits of SDN and machine learning capabilities. Before the proposed design is finalized, an attack impact evaluation in an SDN environment will be explored in the next chapter to understand the issue in depth.

2.5.1 Attack Impact Evaluation in SDN

SDN benefits include centralized pledges with convenient management for adaptation to the data centre or internet architecture. However, associated issues, especially those involving high risk security matters, need to be address and solved before it can be fully deployed and operational within the industry. As elaborated in the first chapter, various types of attack and

mitigation approaches have been identified and researchers have tried to use the benefits of SDN to develop optimum solutions. In this research, attack impact evaluation will be done in an SDN environment to analyse the effect on bandwidth capacity that can be achieved and the impact on the response of SLA in web services.

Evaluating the impact on the bandwidth capacity and response time of the network services running inside the SDN managed networks under attack is important. It would be helpful to assess its performance under different magnitudes of attack. Two critical components in any SLA are the availability of a service to the customers and its responsiveness. In this research, various magnitudes of attacks specific to the SDN were carried out and their impact on the performance of network services determined by analysing the achievable bandwidth and the web services responsiveness during an attack.

2.5.2 Security Threats in SDN

Authors of [18] and [3] discussed various vulnerabilities and threats. Due to its immaturity compared to traditional networks, security threats in SDN networks are more challenging. For example, the impact of a compromised SDN controller will affect the entire network, while in traditional networks, the scope of damage is relatively small. Management threats, control place threats and data plane threats are outlined briefly before attack implementation and impact analysis are discussed.

2.5.3 Management Threats

The installation of the controllers, switches and network applications along with their administration and trust management are under the remit of SDN management. Management refers to the administrative privileges in the operational environment, which, if compromised, would lead to a disastrous situation that would threaten the overall operational environment. Administrative configuration and fine tuning faults by the controller could downgrade the performance of the whole network [19]. Such faulty handling by management would create network outage risks to the controller and the resources involved. Malicious, not fully tested SDN applications could also execute various administrative commands that could interfere and interrupt controller functions, making the network malfunction and become unstable. Malicious programs could monopolize resources such as CPU and memory for a long period of time thus degrading the overall performance of other applications. Use of third-party applications for management purposes could also cause damage as they might have unpatched updates that could allow malicious activities to enter the SDN network. The encoded, unpatched vulnerabilities could trigger various types of attack and ultimately disclose administrative privileged information [20].

2.5.4 Control Plane Threats

The network policies along with operational message updates between OpenFlow switches and the SDN controller are examples of areas of control plane threat. Policy contradictions might occur between all the resources involved causing possible inconsistent or unexpected traffic flows caused by

the rebuttal policy. Man-in-the-middle attacks could also happen due to unprotected channels of communication between the switches and the controller. Unencrypted communication activity can be sniffed out by malicious systems and later exploited to compromise the network and by requiring high priority administrative levels for their execution, this could lead to adding rules intending to harm the environment as well as deleting rules intended to protect it.

A faulty or misbehaving switch could also flood the control plane with unnecessary operational messaging packets such as route requests by sending Packet_In messages to the controller, which needs to respond to such requests and being bombarded by requests would consume controller resources and eventually lead to the SDN environment collapsing.

2.5.5 Data Plane Threats

The data plane consists of flow tables inside the switches, the end hosts and the traffic between them. In SDN flows, an unknown destination packet is referred to the controller for a decision about what should be done with it. Also known as a *missed flow table event*, the controller will reply to the switches with new rules to be updated in the switch flow tables. Longer response times are needed for the first packet to be able to be sent to its destination to allow for checking and initialization the communication between the switch and controller. The specific communication mechanism to the SDN environment provides the attacker with valuable finger-printing characteristics that the attacker can take advantage of to plan and create specific attack methods to bypass the checking mechanism. For example, with the finger-printing

information, the attacker can modify the attack mechanism by sending a valid packet first and once the route has been established, the attack process will start. This will bypass the controller due to SDN communication flows.

2.6 Attack Implementation and Impact Analysis

2.6.1 Environment Setup

Figure 2.4 shows the testbed in which the virtual controller, three host machines and OpenFlow switch were setup using Mininet running Ubuntu 18.04 OS with Intel i5 CPU and 3G RAM. The RYU framework acts as the SDN controller, while OVS is used as an OpenFlow-based virtual switch. Connection speed for all hosts in the simulation was set at 1000 Mbps. For this scenario, SECOD application was chosen. SECOD is a useful tool for detecting and handling Distributed Denial of Service (DDoS) attacks in an SDN environment for several reasons:

i. Real-time traffic monitoring: SECOD continuously monitors network traffic in real-time, which allows it to detect anomalies or patterns that may indicate the presence of a DDoS attack.

ii. Rapid response: Once a DDoS attack is detected, SECOD can quickly respond by dynamically adjusting network configurations to mitigate the impact of the attack. This can help to reduce the impact of the attack and prevent it from causing widespread disruption.

iii. Customizable rules: SECOD allows for the creation of customizable rules that can be tailored to the specific needs of an organization. This enables the program to be adapted to the unique characteristics of an

SDN environment, which can help to improve its effectiveness in detecting and handling DDoS attacks.

iv. Automated alerts: SECOD can be configured to send automated alerts or notifications to administrators or security teams when a DDoS attack is detected. This enables organizations to respond quickly to the attack and take steps to prevent it from causing further damage.

By providing real-time monitoring, rapid response, customizable rules, and automated alerts, SECOD can help to enhance network security and protect against the damaging effects of DDoS attacks. SECOD was chosen as the baseline comparison due to above listed capabilities and it was implemented in a native SDN environment.

The components involved in the setup are explained below:

i. RYU Controller: RYU is an open stack and Python-based framework that supports the implementation of SDN.
ii. Mininet: creates a realistic virtual network running real kernel, switch and application codes on a single machine.
iii. Open vSwitch (OVS): A production quality, multilayer virtual switch licensed under the open-source Apache 2.0 license. It is designed to enable massive network automation through program extension, while still supporting standard management interfaces and protocols.
iv. SECOD [21]: a program implemented as an application of the RYU controller using python-based scripts to do the functions needed for detecting and handling DoS attacks. It communicates with the RYU

controller using the northbound interface to process and store different counters as well as push different command instructions to the switch.

v. iPerf: the network bandwidth measurement tool for collecting the maximum bandwidth in the network during the test.

vi. tcping: these are command line tools for collecting web service responses in the network during the test.

vii. Gnuplot: the tools for producing graphs for the statistics being collected.

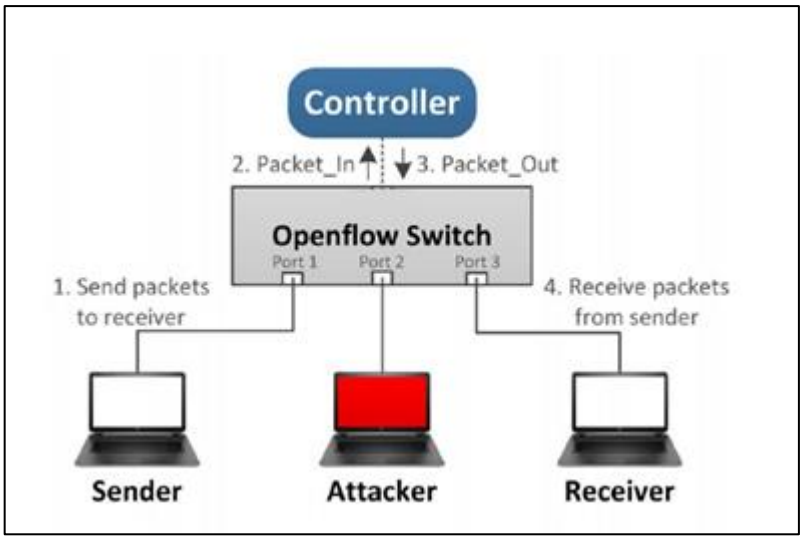


Figure 2.4 : Testbed Setup

All the components are setup in a Mininet environment. The sender of the TCP stream is connected to port 1 of the switch, a DoS attacker is connected to port 2 and a receiver of the TCP stream is connected to port 3. Web service responses will also be collected during the simulation. The SDN controller is installed in Ubuntu OS and connected remotely from the Mininet environment. The bandwidth that can be achieved during the attack and the performance of the web service response will be collected and analysed to evaluate the impact of the attack on the SDN environment.

All three host machines used iPerf applications for gathering traffic generation and statistics. Consistent TCP streaming for 30 seconds in 10 cycles was established between the Sender and the Receiver. After the given period, iPerf reports the average bandwidth achieved during the time period. In the Attacker, a bash script uses iPerf in a loop to continuously send UDP packets with a high frequency to different IP addresses. This allows the rapid creation of new flows in the network, emulating a DoS attack on the control and data plane of the SDN.

Four (4) types of traffic simulation were carried out, as below:

i. Without Attack

   TCP traffic simulation from Sender to Receiver for gathering the bandwidth recorded for network traffic without any DoS attack, for both the normal controller configuration and SECOD being deployed.

ii. Slow Attack

   A TCP traffic simulation from Sender to Receiver for gathering the bandwidth recorded for network traffic during a DoS attack from the Attacker for both with and without SECOD being deployed. The transmission time-to-transmit used for this scenario was 1 second. Higher frequencies are the result of higher volumes of Packet_In messages for the Controller to handle, so DoS attacks happen.

iii. Medium Attack

TCP traffic simulation from Sender to Receiver to gather the bandwidth recorded for network traffic during a DoS attack from Attacker for both with and without SECOD deployed. The transmission time-to-transmit used for this scenario was 0.1 seconds.

iv. Fast Attack

TCP traffic simulation from Sender to Receiver for gathering the bandwidth recorded of network traffic in a DoS attack from an Attacker for both with and without SECOD deployed. The transmission time-to-transmit used for this scenario was 0.001 seconds to emulate the most DoS attack activity in the environment.

2.6.2 Implementation Findings

Without Attack

Both simulations with and without SECOD recorded similar findings as the traffic flows were normal and no DoS attack activity occurred. Bandwidth readings and the graph shown in Figure 2.5 for 10 runs were recorded and summarised in Table 2.3 below:

Without SECOD

Test Number	Mean Bandwidth (Mbps)
1	938
2	938
3	944
4	944
5	926
6	946
7	947
8	946
9	946
10	939

With SECOD

Test Number	Mean Bandwidth (Mbps)
1	932
2	948
3	952
4	948
5	944
6	933
7	942
8	945
9	950
10	952

Table 2.3 : Bandwidth reading for without attack scenario

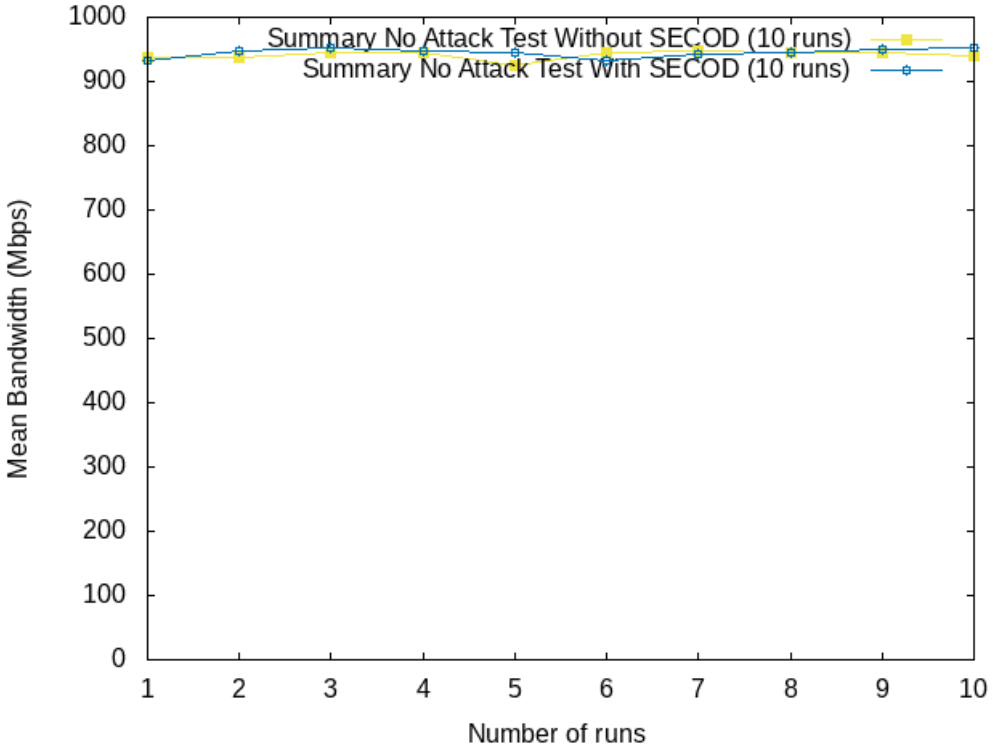


Figure 2.5 : Bandwidth summary without attacks

No significant difference recorded between the simulations and it is assumed that the setup functioned properly when handling traffic test without any DoS attack. The bandwidth that can be achieved is almost 1 Gbps, as the speed setup for the simulation.

Web service responses without any performance test recorded during the test are plotted in Figure 2.6 and Figure 2.7, below. The graph shows the majority are between 4 to 6 milliseconds for web services' responses to the query sent.

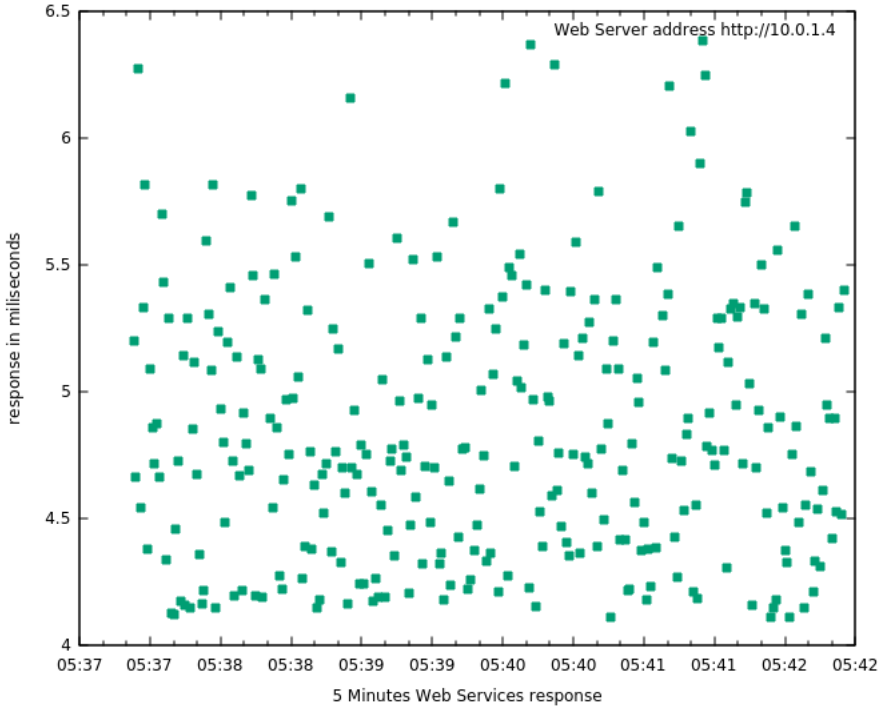


Figure 2.6 : Web Service responses without SECOD

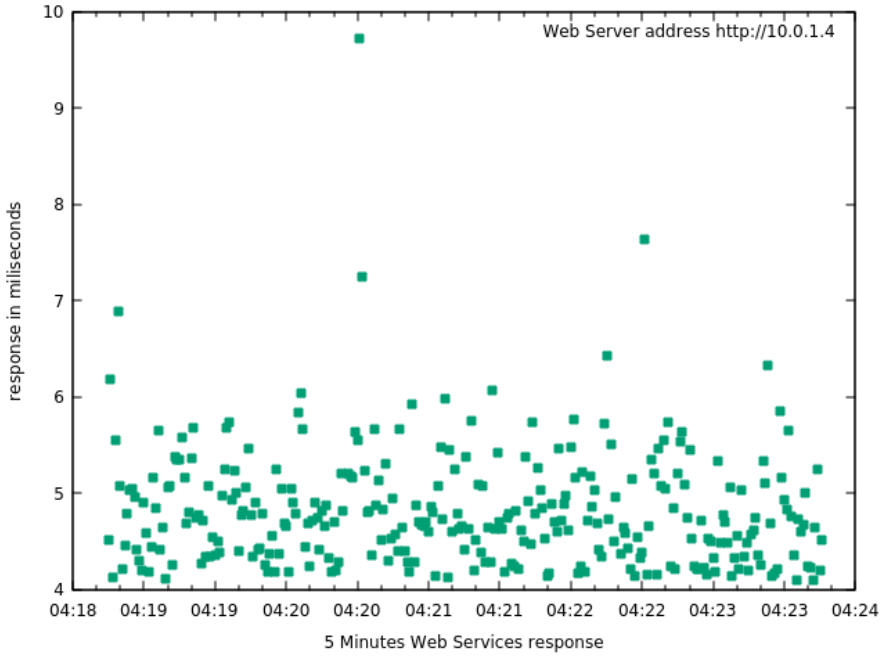


Figure 2.7 : Web Service responses with SECOD

During the performance test, readings between 65 to 75 milliseconds were recorded for web service responses with and without SECOD, as shown in Figure 2.8 and Figure 2.9 : Web Service responses with SECOD during the performance test.

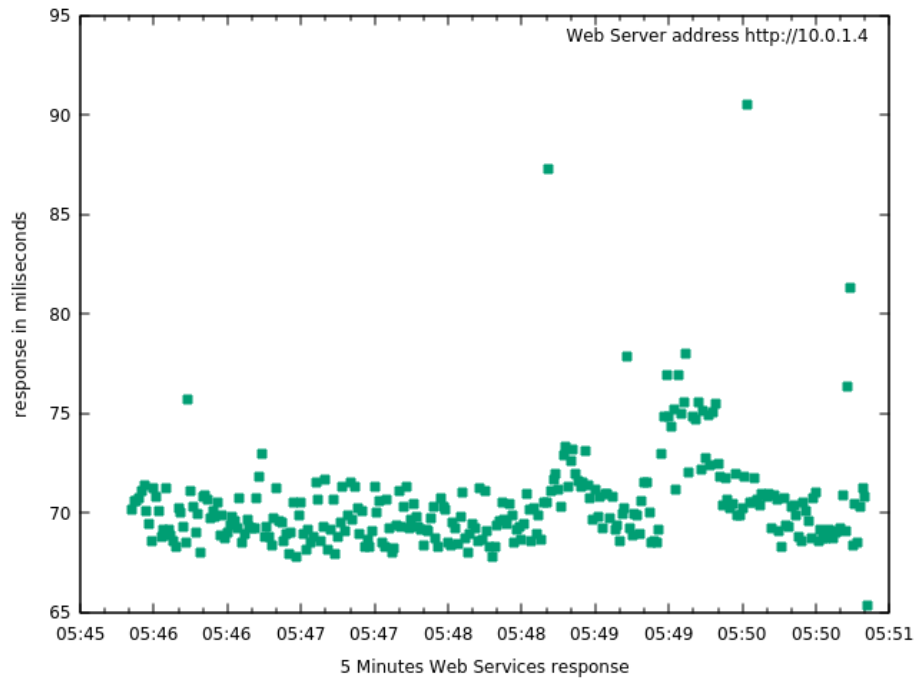


Figure 2.8 : Web Service responses without SECOD during the performance test

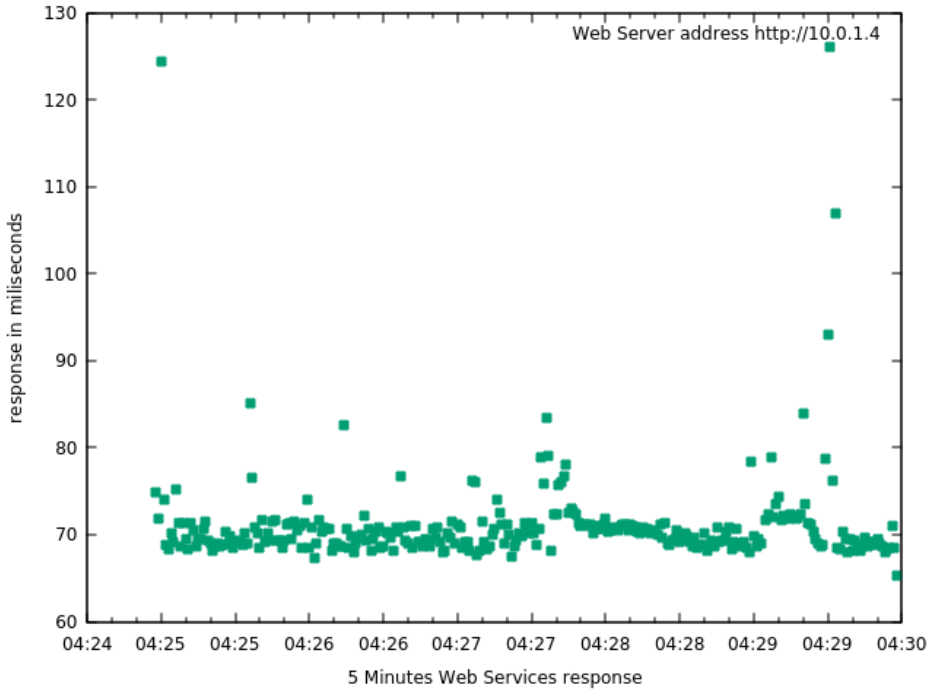


Figure 2.9 : Web Service responses with SECOD during the performance test

Both experiments show the response readings for the web services were quite similar during the performance test.

2.6.3 Slow Attack

Bandwidth reading and the graph for 10 runs were recorded and are summarised in Table 2.4 below:

Without SECOD

Test Number	Mean Bandwidth (Mbps)
1	531
2	531
3	531
4	531
5	532
6	532
7	532
8	531
9	531
10	532

With SECOD

Test Number	Mean Bandwidth (Mbps)
1	531
2	532
3	531
4	530
5	532
6	530
7	532
8	531
9	532
10	529

Table 2.4 : Bandwidth reading for the slow attack scenario

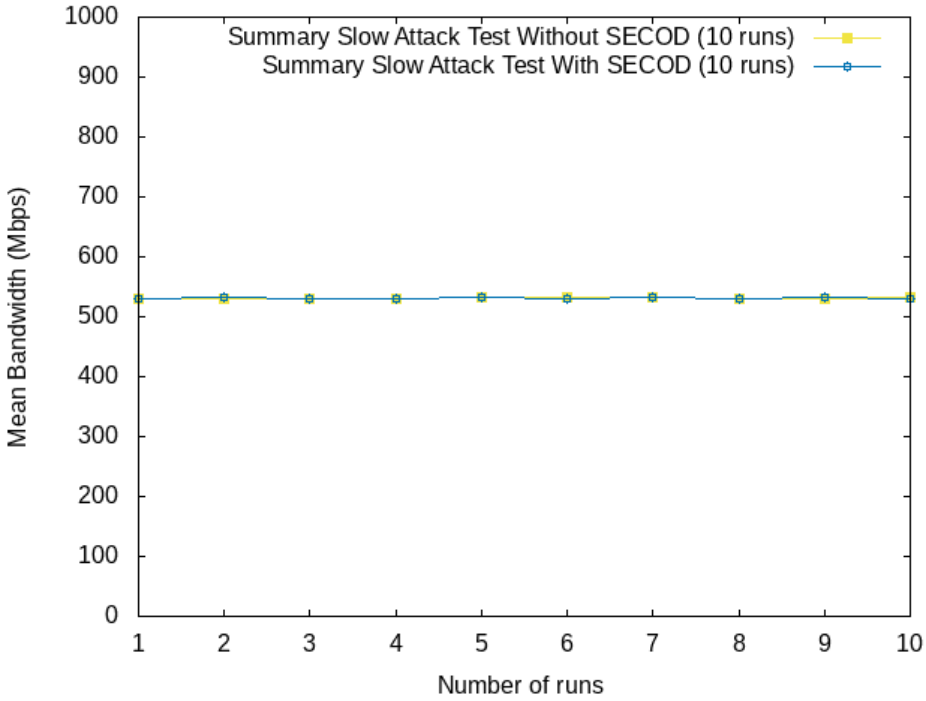


Figure 2.10 : Bandwidth summary during the slow attack test session

Traffic for both tests dropped to half capacity, around 500 Mbps during the slow attack DoS session. With the implementation of SECOD, access from the source of the DDoS dropped but with no significant impact on the traffic bandwidth achieved.

Web service responses during a slow attack session are shown below, in Figure 2.11 and Figure 2.12.

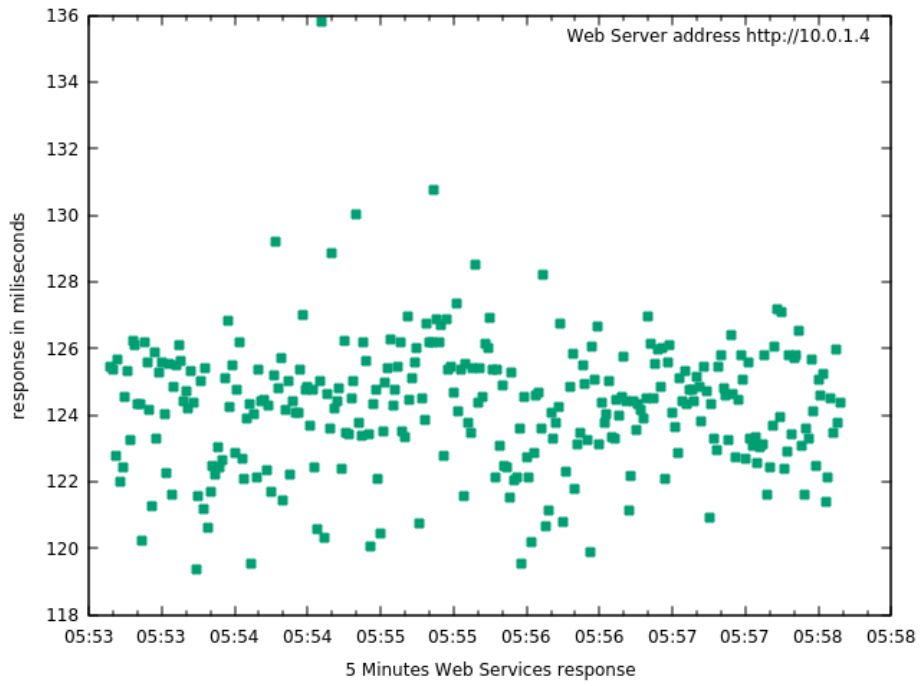


Figure 2.11 : Web Service responses without SECOD during the slow attack

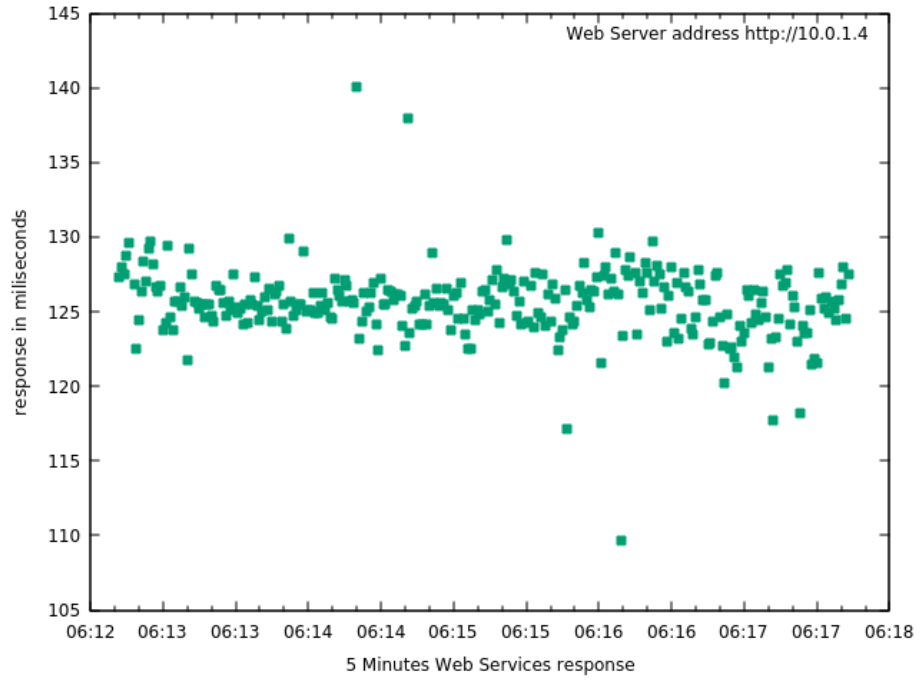


Figure 2.12 : Web Services response with SECOD during the slow attack

Both experiments show the response readings for the web services were similar during the performance test with readings around 120 ms to 130 ms.

The difference is that, during a drop policy enforced in SECOD, the responses are more similar compared to the scattered readings with no policy enforced.

2.6.4 Medium Attack

Bandwidth readings and the graph for 10 runs were recorded and are summarised in Table 2.5 and Figure 2.13 below:

Without SECOD

Test Number	Mean Bandwidth (Mbps)
1	532
2	532
3	532
4	531
5	532
6	532
7	532
8	532
9	532
10	532

With SECOD

Test Number	Mean Bandwidth (Mbps)
1	530
2	531
3	532
4	533
5	530
6	531
7	531
8	532
9	531
10	531

Table 2.5 : Bandwidth reading in a medium attack scenario

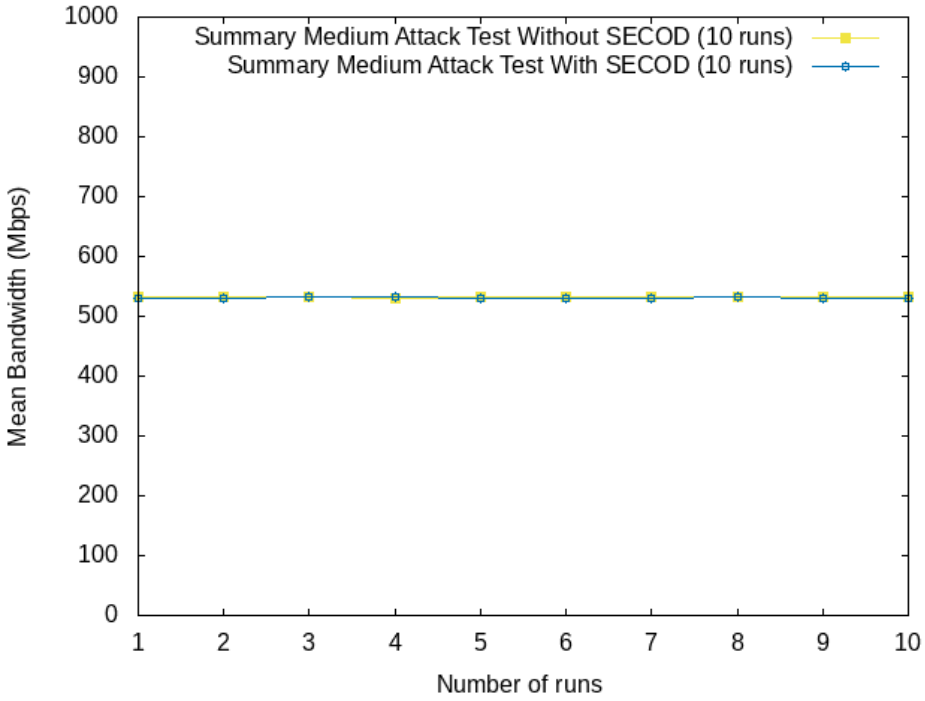


Figure 2.13 : Bandwidth summary during a medium attack session

Traffic for both tests dropped to half capacity, around 500 Mbps during the medium attack DoS session. With the implementation of SECOD, access from the source of the DDoS dropped, but no significant impact r in the traffic bandwidth achieved was recorded.

Web service responses during a medium attack session are shown below in Figure 2.14 and Figure 2.15.

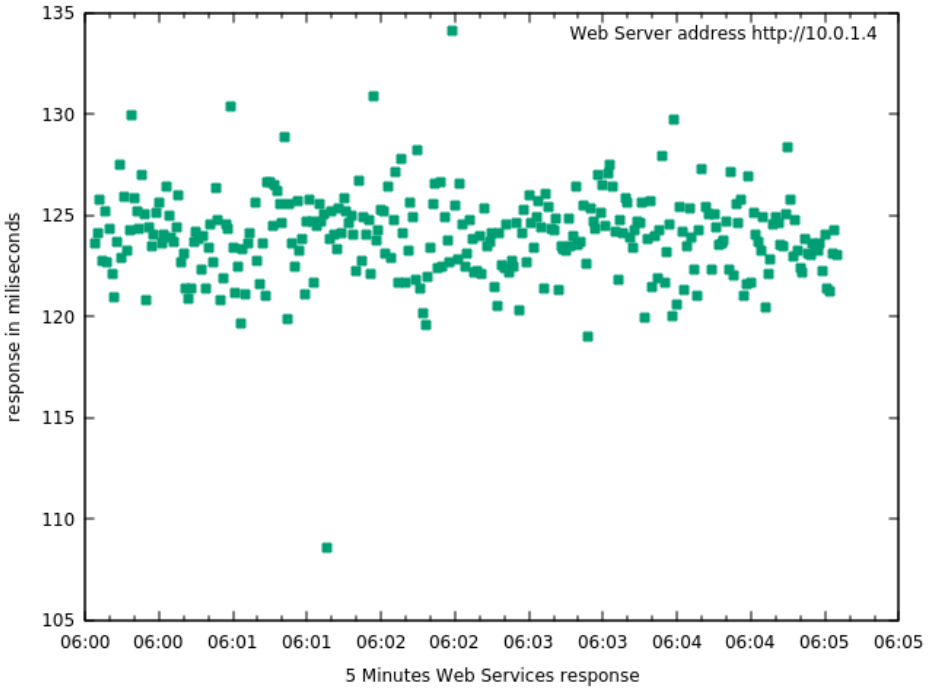


Figure 2.14 : Web Service responses without SECOD during the medium attack

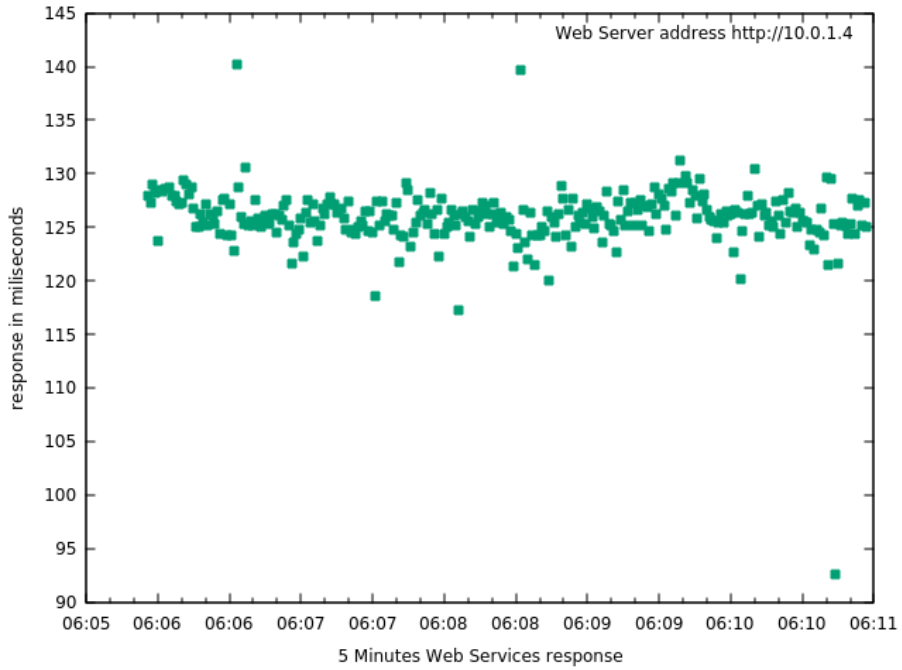


Figure 2.15 : Web Services response with SECOD during the medium attack

2.6.5 Fast Attack

Bandwidth readings and the graph for 10 runs were recorded and summarised below in Table 2.6 and Figure 2.16:

Without SECOD

Test Number	Mean Bandwidth (Mbps)
1	532
2	533
3	531
4	532
5	533
6	534
7	532
8	532
9	534
10	533

With SECOD

Test Number	Mean Bandwidth (Mbps)
1	530
2	534
3	536
4	531
5	534
6	531
7	533
8	531
9	531
10	531

Table 2.6 : Bandwidth readings for the fast attack scenario

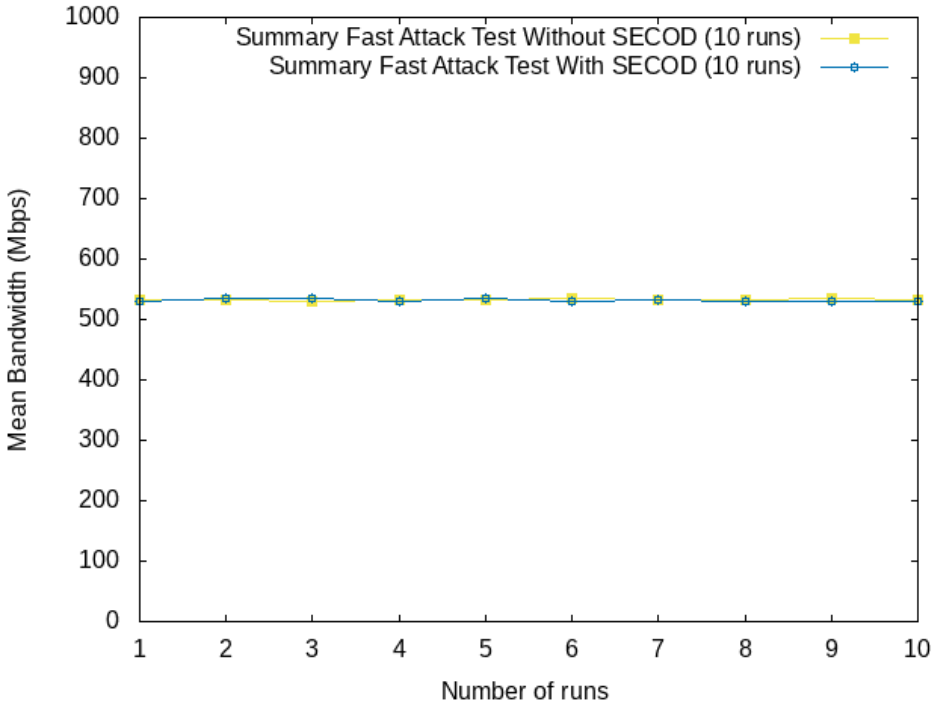


Figure 2.16 : Bandwidth summary of the fast attack session

Traffic for both tests dropped to half capacity, around 500 Mbps during fast attack DoS session. With the implementation of SECOD, access from the source of the DDoS dropped but no significant impact was recorded on the traffic bandwidth achieved.

Web service responses during the fast attack session shown below in Figure 2.17 and Figure 2.18.

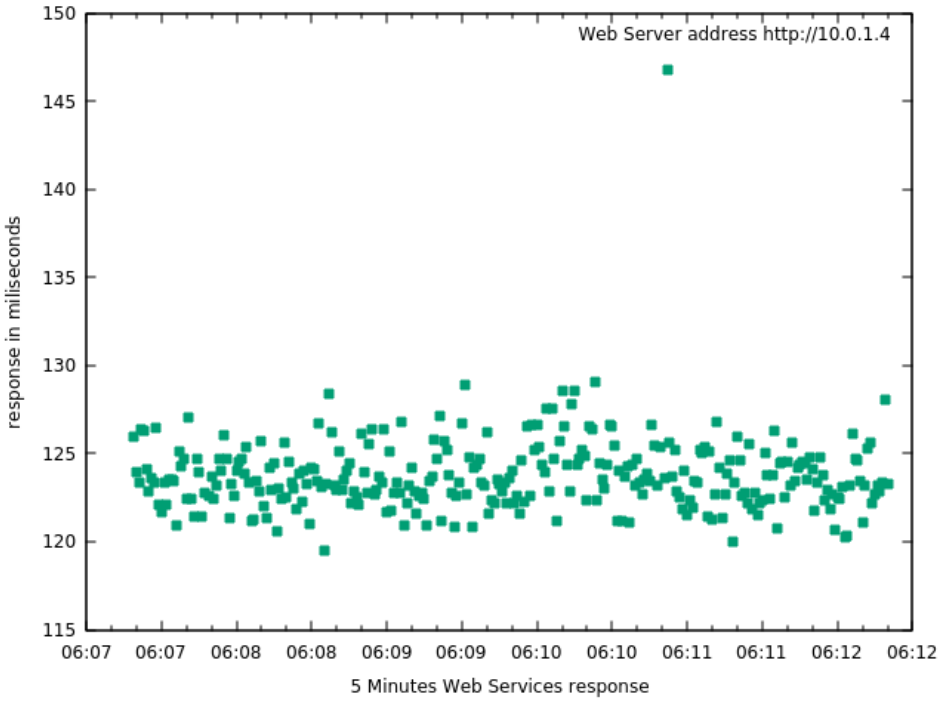


Figure 2.17 : Web Service responses without SECOD during the fast attack

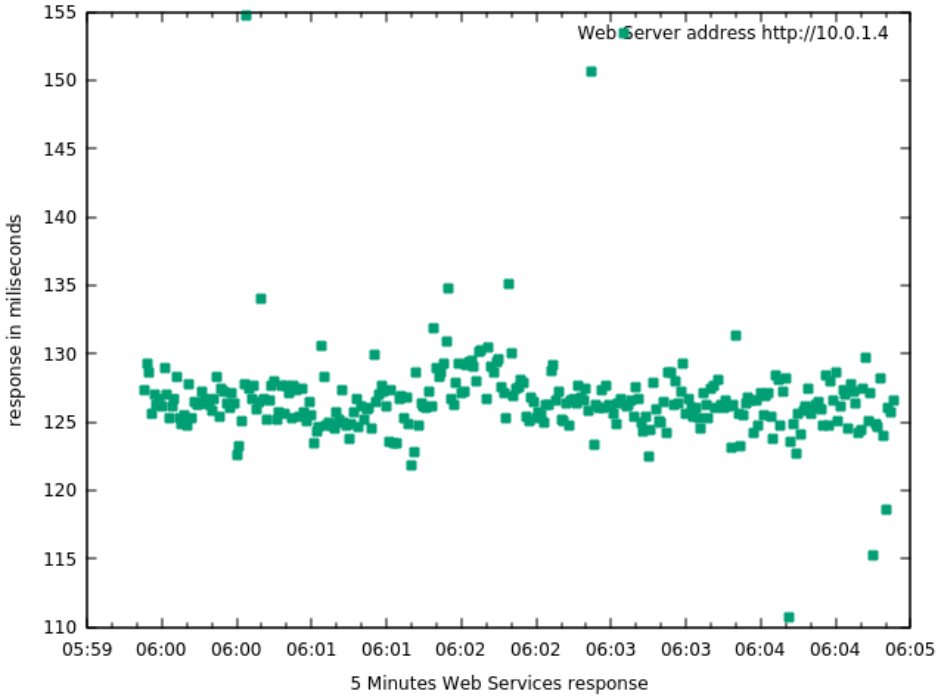


Figure 2.18 : Web Service responses with SECOD during the fast attack

2.6.6 Simulation Summary

The reduction in available bandwidth and higher response rate from web services has shown a significant impact was caused by the attack in the SDN environment. The objective of analysing the findings and becoming familiar with SDN traffic handling has been achieved from the simulation. Although no significant impact was recorded when comparing results during all the attack scenarios, valuable information was gathered for understanding how to use SDN traffic information. Collecting traffic data, computing traffic use in comparison with the threshold setup, and the pushing policy to the OpenFlow switch experienced in the simulation have all provided good basic knowledge for exploring attack handling in an SDN environment in future research. A mitigation plan for improving the bandwidth during an attack and maintaining a low response time for the web services will be proposed in the next chapter.

Before a good machine learning solution can be proposed, it is important to understand the process of selecting features from the available dataset and the optimum classification mechanism for analysing and detecting attacks related to an SDN environment. Chapter three will cover the research related to attack classification using machine learning capabilities.

2.7 Comparison of Machine Learning attacks as classified in an SDN environment

ML is a type of AI that involves using algorithms and statistical models to enable computer systems to learn from data and make decisions or predictions without being explicitly programmed to do so. In the context of SDN, ML can be used to classify different types of network traffic, such as

video streaming, web browsing, or file sharing. This information can be used to optimize network performance and prioritize certain types of traffic over others. ML algorithms also can analyse network traffic patterns and identify anomalous behaviour that may indicate a security threat, such as a DDoS attack. Optimizing resource allocation in an SDN environment, such as allocating bandwidth or storage resources to different applications or users based on their needs and usage patterns can be adopt. Furthermore, ML are able to predict equipment failures or performance degradation in network devices, allowing for proactive maintenance and minimizing downtime.

Dataset is an important element in any research. Some public access datasets are shared by researchers and contributing agencies as benchmarks to enable researchers to develop models or system to compare their performances with that of others using the same dataset. NSL-KDD is one of the most popular, being shared by the Canadian Institute for Cybersecurity dataset for intrusion and attack detection. The dataset is an updated version of the earlier KDD Cup 99 dataset, aiming to overcome some limitations in that earlier version. Chapter two covers the scope of applying the machine learning approach for attack classification in an SDN simulation. The NSL-KDD dataset is used as the input while a machine learning algorithm is used as the classification mechanism for detecting attacks in SDN. The feature selection and classification is carried out using the Weka tool to capture the performance and compare various algorithms to find the best combination to apply in any future adaptive attack mitigation proposal.

2.7.1 NSL-KDD Dataset

The KDD Cup 99 dataset has been used as a benchmark dataset for many years in intrusion detection development and evaluation. The drawback in this dataset is that it has many redundant records in the training and test data. It was observed that the training and testing data have almost 78% and 75% redundant records respectively. The detection mechanism developed using this dataset become biased towards the classification of frequently found attack records and produce poor classification results for the less frequent, but harmful records due to this redundancy. In this case, the comparison and evaluation of different NIDSs become difficult, as they all produce excellent results on this dataset.

The NSL-KDD came into existence to overcome the limitations of the KDD Cup 99 dataset but is derived from the KDD Cup 99 dataset. The following approaches were taken in the NSL-KDD dataset to improve the KDD Cup 99 dataset. Firstly, all the redundant records from the training and test data in the KDD Cup 99 dataset were removed. After removing redundant records from the KDD Cup 99 dataset, the remaining distinct records were divided into 21 sets based on their classification accuracy by different learning algorithms. Each set contained records that could be correctly classified by a specific number of learning algorithms. This was done to create a more diverse and challenging dataset that could better represent real-world network traffic and attacks.. A record was kept in a set with the same number as the classifiers that accurately classified the record. The records were sampled from each set in a fraction inversely proportional to the fraction of the records in that set over the total number of records in all the sets. Each record in the NSL-KDD

dataset has 41 features including three nominal, four binary, and 34 continuous features along with a label for normal or a particular kind of attack.

NSL-KDD dataset feature list and its description are as shown in Table 2.7 below.

Feature name	Feature Number	Description
duration	1	Length of the connection in seconds
Protocol-type	2	Type of connection protocol
service	3	Destination of network services
flag	4	Status of the connection (normal or error)
src_bytes	5	Source to destination data byte numbers
dst_bytes	6	Destination to source data byte numbers
land	7	1 if the connection is from/to the same host/port; 0 otherwise
wrong_fragment	8	Number of wrong fragments
urgent	9	Number of urgent packets
hot	10	Number of hot indicators
failed_logins	11	Number of failed login attempts
logged_in	12	1 if successfully logged in; 0 otherwise
_compromised	13	Number of compromised conditions
root_shell	14	1 if root shell is obtained; 0 otherwise
su_attempts	15	1 if ‘‘su root’’ command attempted; 0 otherwise
_roots	16	Number of root accesses
_file_creations	17	Number of file creation operations

_shells	18	Number of shell prompts
_access_files	19	Number of operations on access control files
_outbound_cmds	20	Number of outbound commands
is_hot_login	21	1 if the login belongs to the hot list; 0 otherwise
is_guest_login	22	1 if the login is a guest login; 0 otherwise
count	23	Number of connections to the same host as the current connection in the past 2 seconds
srv_count	24	Number of connections to the same service as the current connection in the past 2 seconds
serror_rate	25	% of connections that have ‘‘SYN’’ errors
srv_serror_rate	26	% of connections that have ‘‘SYN’’ errors
rerror_rate	27	% of connections that have ‘‘REJ’’ errors
srv_rerror_rate	28	% of connections that have ‘‘REJ’’ errors
same_srv_rate	29	% of connections to the same service
diff_srv_rate	30	% of connections to different services
srv_diff_host_rate	31	% of connections to different hosts
dst_host_count	32	Count of connections having the same destination host
dst_host_srv_count	33	Count of connections having the same destination host and using the same service
dst_host_same_srv_rate	34	% of connections having the same destination host and using the same service

dst_host_diff_srv_rate	35	% of different services on the current host
dst_host_same_src_port_rate	36	% of connections to the current host having the same source port
dst_host_src_diff_host_rate	37	% of connections to the same service coming from different hosts
dst_host_serror_rate	38	% of connections to the current host that have an S0 error
dst_host_srv_serror_rate	39	% of connections to the current host and specified service that have an S0 error
dst_host_rerror_rate	40	% of connections to the current host that have an RST error
dst_host_srv_rerror_rate	41	% of connections to the current host and specified service that have an RST error

Table 2.7 : NSL-KDD dataset features and their descriptions

The NSL-KDD dataset is commonly used in research related to IDS and network security. One of the applications of IDS is to detect and prevent attacks in SDN environments. The dataset has been widely used in these types of studies because it provides a comprehensive collection of network traffic data that simulates various types of attacks. By using the NSL-KDD dataset to train and evaluate machine learning algorithms or IDS techniques, researchers can assess the effectiveness of these methods in detecting and preventing attacks in SDN environments. Top of Form The NSL-KDD dataset can be beneficial in evaluating the effectiveness of various security mechanisms and techniques in SDN environments. Benefits of using the NSL-KDD dataset in SDN environments is as follows:

i. Realistic data: The NSL-KDD dataset provides a realistic and diverse set of network traffic data that simulates various types of attacks, such as DoS, Probe, R2L, and U2R attacks. This can help researchers to evaluate the effectiveness of different security mechanisms and techniques in detecting and preventing attacks in SDN environments.

ii. Comprehensive feature set: The NSL-KDD dataset includes a comprehensive set of pre-defined features that can be used to analyse network traffic and identify patterns that may indicate an attack. This can help researchers to develop effective machine learning algorithms or intrusion detection systems that can detect and prevent attacks in SDN environments.

iii. Benchmark dataset: The NSL-KDD dataset is widely used as a benchmark dataset for evaluating the performance of intrusion detection systems and related techniques. This can help researchers to compare the performance of different security mechanisms and techniques in detecting and preventing attacks in SDN environments.

iv. Cost-effective: The NSL-KDD dataset is freely available and can be used for research purposes without any cost. This can be beneficial for researchers who may not have access to real-world network traffic data or who may not have the resources to collect and analyse their own data.

2.7.2 Dataset Feature Selection

Previous studies have been made by researchers using the NSL-KDD dataset as the input for machine learning classification. With a combination of proposed classification techniques, different feature selections have been presented to assess the proposed intrusion detection procedure. A summary of related work with similar objectives mentioned earlier is shown in Table 2.8 below.

Ref.	Year.	Number of feature selection	Feature list
[7]	2018	6	1, 2, 5, 6, 24, 36
[22]	2018	6	3, 4, 5, 6, 12, 26
[23]	2013	11	3, 5, 6, 10, 17, 18, 32, 33, 35, 36, 38
[24]	2015	11	3, 4, 12, 26, 29, 30, 31, 32, 33, 38, 39
[25]	2012	11	1, 3, 6, 12, 22, 23, 24, 25, 28, 31, 32
[26]	2104	13	not specified
[27]	2013	13	not specified
[28]	2013	22	not specified
[29]	2010	23	not specified
[30]	2013	27	1, 2, 3, 4, 5, 6, 10, 11, 23, 24, 25, 26, 27, 28, 29, 30, 31, 31 ,33 ,34, 35, 36, 37, 38, 39, 40, 41

Table 2.8 : Dataset selections of previous studies

Feature selection is an important step in building machine learning models and intrusion detection systems (IDS) using the NSL-KDD dataset. The feature selection process involves selecting a subset of relevant features from

the dataset that can be used to build effective models for detecting and preventing attacks. The NSL-KDD dataset originally included 41 features, which were pre-defined by the creators of the dataset. However, several studies have shown that not all of these features are relevant or useful for building effective intrusion detection systems. In fact, using all 41 features can lead to overfitting and reduced model performance. To address this issue, researchers have proposed various methods for feature selection in the NSL-KDD dataset. These methods aim to identify the most relevant and informative features that can be used to build effective IDS models. Some of the popular feature selection methods used in the NSL-KDD dataset include:

i. Correlation-based feature selection: This method involves selecting features that are highly correlated with the target variable (i.e., attack or normal traffic) and removing features that are highly correlated with each other. This can help to reduce redundancy in the feature set and improve model performance.

ii. Recursive feature elimination: This method involves recursively removing the least important features from the dataset and evaluating the model performance after each iteration. This can help to identify the most important features that contribute the most to the model performance.

iii. Principal component analysis: This method involves transforming the original feature set into a lower-dimensional feature space that retains the most important information from the original features. This can help to reduce the dimensionality of the feature set and improve model performance.

Feature selection process in the NSL-KDD dataset has evolved over time as researchers have identified the most relevant and informative features for building effective IDS models. The goal of feature selection is to reduce the dimensionality of the feature set while retaining the most important information, which can help to improve the accuracy and performance of the IDS models. The number of features selected has tended to decrease in recent years because selecting fewer features makes the classification process much faster. Although fewer features are selected, the accuracy of the proposed model needs to be high enough to yield good results. The aim of analysing different projects' feature selection from the same dataset is to propose alternative selection features that can yield a better result.

2.8 Simulation Setup

Feature selection and classification methods are available in built-in WEKA, a machine learning work bench. As proposed by [22], 4 feature selection methods have been chosen, namely CfsSubsetEval, GainRatioAttributeEval, OneRAttributeEval and SymmetricalUncertAttributeEval. The feature selections in WEKA are shown in Figure 2.19 below.

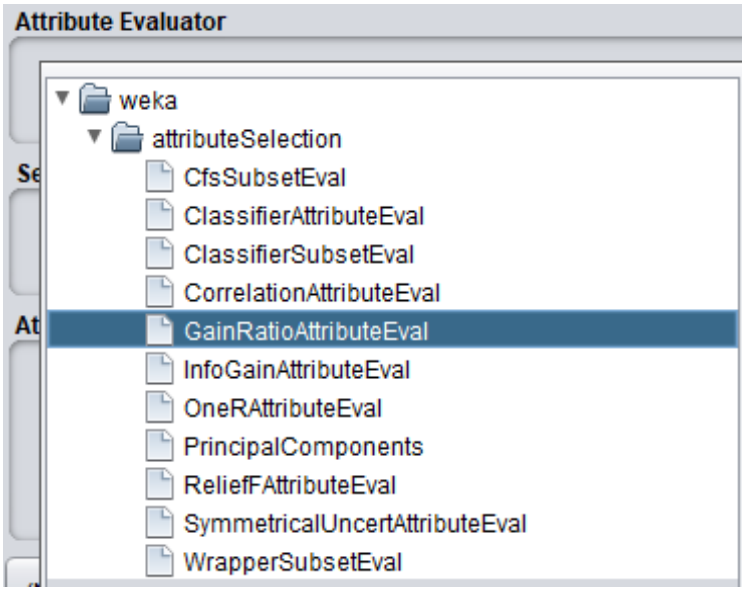


Figure 2.19 : WEKA attribute selection

Classification using the available algorithm in Weka are shown in Figure 2.20 below. For this work's comparisons, 4 classifiers have been chosen, namely BayesNet, Logistics, NBTree and IB1.

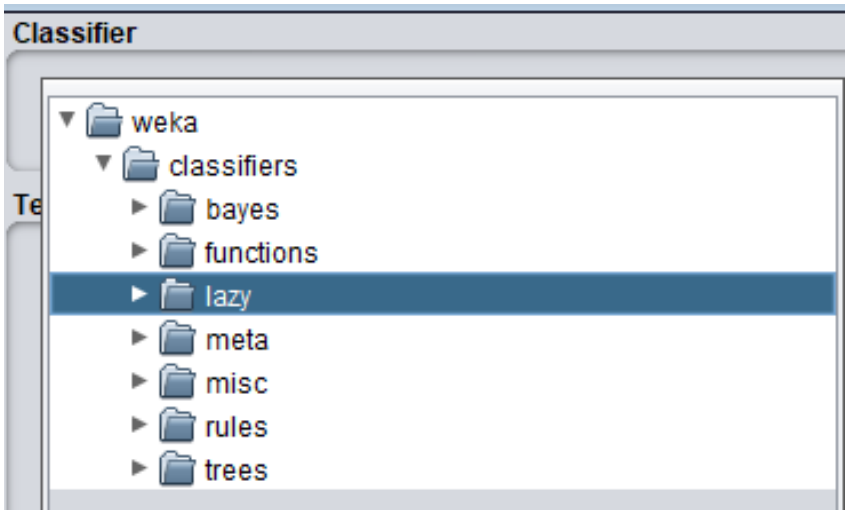


Figure 2.20 : WEKA classification selection

To select the best combination of feature selection methods for future research in machine learning deployment, two previous research projects that utilized a 6-feature selection approach were identified and selected. The

purpose of selecting these two projects was to compare their feature selection methods and identify the most effective combination for future research. Alternative 1 and Alternative 2 feature selections were suggested as in list in Table 2.9 below. The comparison of the evaluation measurements discussed in the findings section.

Ref.	Year.	Number of feature selection	Feature list
Full Features		41	All features
[7]	2018	6	1, 2, 5, 6, 24, 36
[22]	2018	6	3, 4, 5, 6, 12, 26
Alternative 1		6	3, 4, 5, 12, 25, 26
Alternative 2		6	3, 4, 5, 6, 12, 25

Table 2.9 : Feature selection options

2.9 Findings

Results from the NSL-KDD test using selected features are shown in Table 2.10 below.

Evaluation Measurement	Dataset	Classifier			
		BayesNet	Logistic	NBTree	IB1
Detection Rate	Full Features	74.43	75.60	79.68	79.35
	Ref 19	77.62	50.12	79.21	79.43
	Ref 30	78.65	77.84	82.98	82.49
	Alternative 1	78.73	77.86	**83.13**	82.29
	Alternative 2	78.68	77.65	**83.46**	81.99
False Alarm	Full Features	0.200	0.203	0.175	0.165
	Ref 19	0.177	0.381	0.165	0.166
	Ref 30	0.169	0.192	0.150	0.154
	Alternative 1	0.168	0.191	**0.148**	0.156
	Alternative 2	0.169	0.193	**0.146**	0.158
Precision	Full Features	0.822	0.804	0.824	0.841
	Ref 19	0.836	0.728	0.843	0.839
	Ref 30	0.841	0.808	0.847	0.842
	Alternative 1	0.842	0.808	**0.848**	0.840
	Alternative 2	0.841	0.807	**0.850**	0.838
Recall	Full Features	0.744	0.756	0.797	0.794
	Ref 19	0.776	0.501	0.792	0.794
	Ref 30	0.787	0.778	0.830	0.825
	Alternative 1	0.787	0.779	0.831	0.823
	Alternative 2	0.787	0.777	0.835	0.820

F-Measure	Full Features	0.739	0.754	0.797	0.792
	Ref 19	0.774	0.405	0.791	0.793
	Ref 30	0.785	0.778	0.831	0.826
	Alternative 1	0.785	0.779	0.832	0.824
	Alternative 2	0.785	0.777	0.835	0.821

Table 2.10 : Selected features test results

The detection rate evaluation using the NBTree classifier, Alternative 1 and Alternative 2 feature selections show a slight increase in the results achieved, with readings of 83.13% and 83.46% detection rates, as shown in Figure 2.21 and Figure 2.22 respectively using the test dataset. The full feature selection level was 79.68%, as shown in Figure 2.23 while reference [7] achieved a 79.21% level, as shown in Figure 2.24 and ref [22] achieved 82.98%, as shown in Figure 2.25.

```
Attribute mappings:

Model attributes                    Incoming attributes
----------------------------        ----------------
(nominal) service               --> 3 (nominal) service
(nominal) flag                  --> 4 (nominal) flag
(numeric) src_bytes             --> 5 (numeric) src_bytes
(nominal) logged_in             --> 12 (nominal) logged_in
(numeric) serror_rate           --> 25 (numeric) serror_rate
(numeric) srv_serror_rate       --> 26 (numeric) srv_serror_rate
(nominal) class                 --> 42 (nominal) class


Time taken to build model: 9.64 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 90.67 seconds

=== Summary ===

Correctly Classified Instances       18742               83.1352 %
Incorrectly Classified Instances      3802               16.8648 %
Kappa statistic                          0.6651
Mean absolute error                      0.1869
Root mean squared error                  0.4139
Relative absolute error                 37.02   %
Root relative squared error             81.8081 %
Total Number of Instances            22544

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.915    0.232    0.749      0.915    0.824      0.677    0.939     0.941     normal
                 0.768    0.085    0.923      0.768    0.838      0.677    0.939     0.942     anomaly
Weighted Avg.    0.831    0.148    0.848      0.831    0.832      0.677    0.939     0.941

=== Confusion Matrix ===

    a    b   <-- classified as
 8883  828 |    a = normal
 2974 9859 |    b = anomaly
```

Figure 2.21 : Full feature selection

```
Attribute mappings:

Model attributes             Incoming attributes
-----------------------      ----------------
(nominal) service        --> 3 (nominal) service
(nominal) flag           --> 4 (nominal) flag
(numeric) src_bytes      --> 5 (numeric) src_bytes
(numeric) dst_bytes      --> 6 (numeric) dst_bytes
(nominal) logged_in      --> 12 (nominal) logged_in
(numeric) serror_rate    --> 25 (numeric) serror_rate
(nominal) class          --> 42 (nominal) class


Time taken to build model: 9.54 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 38.13 seconds

=== Summary ===

Correctly Classified Instances       18817               83.4679 %
Incorrectly Classified Instances      3727               16.5321 %
Kappa statistic                          0.6714
Mean absolute error                      0.1649
Root mean squared error                  0.386
Relative absolute error                 32.6749 %
Root relative squared error             76.2864 %
Total Number of Instances            22544

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.914    0.225    0.754      0.914    0.826      0.683    0.951     0.950     normal
                 0.775    0.086    0.923      0.775    0.842      0.683    0.951     0.957     anomaly
Weighted Avg.    0.835    0.146    0.850      0.835    0.835      0.683    0.951     0.954

=== Confusion Matrix ===

    a    b   <-- classified as
 8876  835 |    a = normal
 2892 9941 |    b = anomaly
```

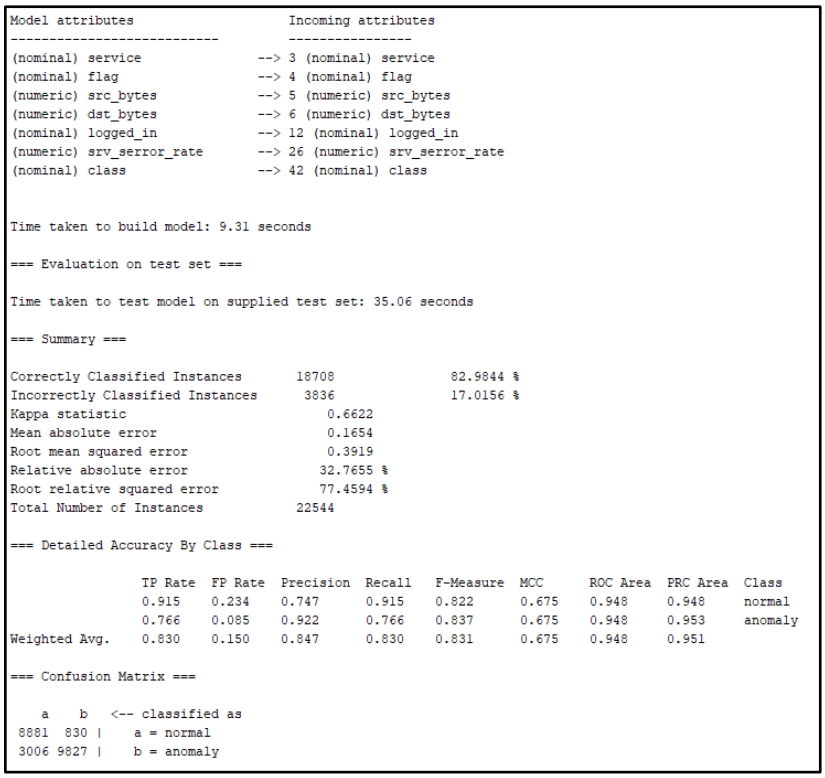
Figure 2.22 : Reference [19] feature selection

```
Attribute mappings:

Model attributes                          Incoming attributes
----------------                          ----------------
(numeric) duration                     --> 1 (numeric) duration
(nominal) protocol_type                --> 2 (nominal) protocol_type
(numeric) src_bytes                    --> 5 (numeric) src_bytes
(numeric) dst_bytes                    --> 6 (numeric) dst_bytes
(numeric) srv_count                    --> 24 (numeric) srv_count
(numeric) dst_host_same_src_port_rate  --> 36 (numeric) dst_host_same_src_port_rate
(nominal) class                        --> 42 (nominal) class


Time taken to build model: 21 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 36.87 seconds

=== Summary ===

Correctly Classified Instances       17858               79.214  %
Incorrectly Classified Instances      4686               20.786  %
Kappa statistic                          0.5967
Mean absolute error                      0.2053
Root mean squared error                  0.4393
Relative absolute error                 40.6675 %
Root relative squared error             86.8254 %
Total Number of Instances            22544

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.969    0.341    0.682      0.969    0.801      0.637    0.949     0.938     normal
               0.659    0.031    0.965      0.659    0.783      0.637    0.949     0.950     anomaly
Weighted Avg.  0.792    0.165    0.843      0.792    0.791      0.637    0.949     0.945

=== Confusion Matrix ===

    a    b   <-- classified as
 9407  304 |    a = normal
 4382 8451 |    b = anomaly
```

Figure 2.23 : Ref [27] feature selection

```
Time taken to build model: 331.05 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.43 seconds

=== Summary ===

Correctly Classified Instances       17964               79.6842 %
Incorrectly Classified Instances      4580               20.3158 %
Kappa statistic                          0.6002
Mean absolute error                      0.2
Root mean squared error                  0.4404
Relative absolute error                 39.6274 %
Root relative squared error             87.0494 %
Total Number of Instances            22544

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.913    0.291    0.704      0.913    0.795      0.620    0.938     0.928     normal
               0.709    0.087    0.915      0.709    0.799      0.620    0.938     0.949     anomaly
Weighted Avg.  0.797    0.175    0.824      0.797    0.797      0.620    0.938     0.940

=== Confusion Matrix ===

    a    b   <-- classified as
 8868  843 |    a = normal
 3737 9096 |    b = anomaly
```

Figure 2.24 : Alternative 1 feature selection



```
Model attributes                    Incoming attributes
----------------------------        ----------------
(nominal) service               --> 3 (nominal) service
(nominal) flag                  --> 4 (nominal) flag
(numeric) src_bytes             --> 5 (numeric) src_bytes
(numeric) dst_bytes             --> 6 (numeric) dst_bytes
(nominal) logged_in             --> 12 (nominal) logged_in
(numeric) srv_serror_rate       --> 26 (numeric) srv_serror_rate
(nominal) class                 --> 42 (nominal) class


Time taken to build model: 9.31 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 35.06 seconds

=== Summary ===

Correctly Classified Instances       18708               82.9844 %
Incorrectly Classified Instances      3836               17.0156 %
Kappa statistic                          0.6622
Mean absolute error                      0.1654
Root mean squared error                  0.3919
Relative absolute error                 32.7655 %
Root relative squared error             77.4594 %
Total Number of Instances            22544

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.915    0.234    0.747      0.915    0.822      0.675    0.948     0.948     normal
                 0.766    0.085    0.922      0.766    0.837      0.675    0.948     0.953     anomaly
Weighted Avg.    0.830    0.150    0.847      0.830    0.831      0.675    0.948     0.951

=== Confusion Matrix ===

    a    b   <-- classified as
 8881  830 |    a = normal
 3006 9827 |    b = anomaly
```

Figure 2.25 : Alternative 2 feature selection

The proposed system design involves the use of adaptive attacks and feature selection to improve the accuracy of intrusion detection systems. Two alternative feature selection methods were considered in the study, and it was found that Alternative 2 feature selection yielded better results than the other method. Alternative 2 involves selecting a set of features that are highly correlated with the target variable and have low correlation with each other.

After the proposed adaptive attack has been deployed, the selected features are used to prompt mitigation actions. Specifically, the feature list is used to identify the type and characteristics of the attack, which in turn informs the selection of an appropriate mitigation strategy. For example, if the feature list indicates that the attack is a denial of service (DoS) attack, the system can initiate a traffic filtering mechanism to block traffic from the attacker's IP address. Alternatively, if the attack is a port scan, the system can modify

firewall rules to block traffic from the attacker's source port. Overall, the proposed system design demonstrates the effectiveness of feature selection in improving the accuracy of intrusion detection systems and the importance of using selected features to prompt mitigation actions in response to adaptive attacks. The use of Alternative 2 feature selection method ensures that the selected features are highly relevant to the target variable and minimizes the possibility of selecting features that are highly correlated with each other, which could lead to overfitting and poor generalization performance of the system.

2.10 Summary

In this chapter, we discussed in detail the SDN architecture, OpenFlow protocol, ML, DL and also the used datasets. This chapter also provided an overview of the DDoS attack and some related work in SDNs. The attacks implementation and simulation in this chapter shows the impact in an SDN architecture. We describe and explain the different types of attack magnitude that impacted the overall performance of the SDN environment. At the end of this chapter, comparison of ML algorithm being use to classify the attack in SDN traffic to show the flexibility of the SDN paradigm in monitoring and detecting network attacks.

CHAPTER 3 : Native SDN Intrusion Detection using Machine Learning

3.1 Introduction

In recent years, SDN has been widely studied and put into practice to assist in network management especially with regards to newly evolved network security challenges. SDN decouples the data and control planes, while maintaining a centralised and global view of the whole network. However, the separation of control and data planes makes SDN vulnerable to security threats. Seven main categories directly related to these risks have been identified which are unauthorized access, data leakage, data modification, compromised application, denial of services (DoS), configuration issue and system-level SDN security. One example of the most concerning issue is related to Distributed denial-of-service (DDoS). DDoS attacks have been a real threat for network, digital and cyber infrastructure. The magnitude of disruptions are massive and can bring down ICT infrastructure services. Various reasons for attack can be synopsized with the main objective of DDoS attack as to paralyze network services by bombarding illegitimate traffic to targeted servers, network links or network devices [31]. Another example is that attackers can oversee and falsify network management information, implement saturation attack, become a perpetrator in man-in- the-middle attacks and so on. Therefore, it is important to adopt a defense mechanism for securing SDN architecture by analyzing vulnerability and improve trust management in traffic handling. One of the proposed solutions for detecting such intrusion is the usage of the intrusion detection system (IDS). There are 2 types of IDS which are signature-based IDS and anomaly detection based

IDS. The first type of IDS depends on the pre-identified signature of an intrusion that has been known before whilst the second type of IDS observed the traffic that deviates so much from normal and deemed suspicious or abnormal. Efficient approach in providing a high level of security in areas such as access control, authentication, malicious outbreak detection and others has resulted by adapting machine learning features and capabilities [32]. Network traffic classification either normal or attack are done via the usage of the Naive Bayes algorithm, Support Vector Machine, K-nearest Neighbor, Neural Network, Recurrent Neural Network, Deep Neural Network and others [33] [6]. Reinforcement learning practicality also being embedded in the case of unlabelled data and the needs of the different approaches for intrusion identification and detection. With the abovementioned advancement and additional capabilities provided by SDN architecture, research of incorporating machine learning has been done to improve network security. Researchers also have embarked autoencoder approach as an alternative in network intrusion detection environments such as the works of [9] and [34]. Autoencoders are unsupervised learning techniques that make use of the neural network for undertaking representation learning. A bottleneck in the network is imposed to force a compressed knowledge from the original input. An ideal autoencoder model can be achieved by balancing the sensitivity of the input for accurate reconstruction build up and take into account overfitting condition which simply memorizing the input data. For most cases, a loss function usually being constructed with the term on encouraging for input sensitivity and secondly by discouraging overfitting or memorization of input data, for example by adding regularizer. Embracement of machine learning and autoencoder such as the works above has outset promising solution in

intrusion detection. Progression in the accuracy level, reducing training and execution time needed can still be improved especially to be embedded in a fast speed of SDN environment. This further improvement motivates in outlining this research. In summary, the contributions of this chapter are the following:

We introduce a hybrid combination of an autoencoder (AE) and random forest (RF) algorithm in the SDN environment. Our AERF approach yields a detection rate of 98.4% using a minimum number of features. We also evaluate the network performance of the proposed approach in the SDN. The test results show that our approach is a significant potential for real-time detection with minimum impact on the controller and noteworthy processing time.

3.2 Architecture

A. SDN-based Intrusion Mitigation Architecture

The proposed architecture is as shown in Figure 3. 1 below.

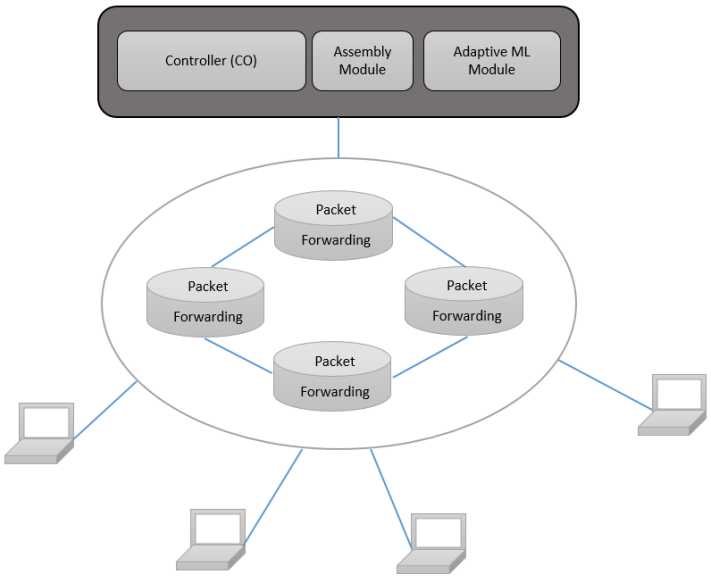


Figure 3. 1 : Intrusion Mitigation Architecture

Two main functionalities are Assembly Module for statistics collection and Adaptive ML Module which will be adopting deep learning for analysing traffic received for intrusion detection and enforcing policies enforcement with capabilities of machine learning in the SDN environment. The proposed solution will focus on mitigating the attack with enrichment of machine learning benefits. The functionality of the main components is presented as follows:

a) Assembly Module:

Collecting traffic information of the SDN network. Collected data will be assembled as needed input for the controller to take proper actions. Utilized standard Openflow protocol which periodically sends an Openflow flow stats request message to all Openflow connected switches. Managed switches will respond back to the message that contains traffic flow statistics.

b) Adaptive ML Module:

Proposed intrusion detection using AERF will be deployed and attack traffic will be penalized when detected by the controller using machine learning functionality in the adaptive ML module.

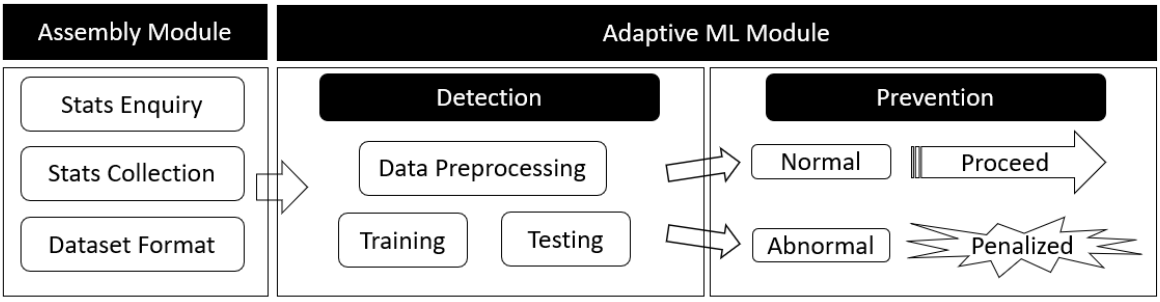


Figure 3. 2 : Intrusion Mitigation Module

The functionality of the proposed module are as shown in Figure 3. 2. Assembly Module will handle the statistics inquiry and collection from all OpenFlow enable switches. The gathered data will then be processed with the required format for the controller to proceed with detection and prevention in the adaptive ML module. The training and testing process of the detection process needs to be fast and yield high accuracy as it will deal with the high

speed data flow. Upon the classifying process, normal traffic will proceed as usual whilst abnormal traffic will be penalized.

3.3 Experimental Methodology

A. Dataset

The NSL-KDD dataset is a popular choice for research and experimentation in the field of network intrusion detection and machine learning. The reasons why the NSL-KDD dataset selected is as below.

i. Standardized Benchmark: The NSL-KDD dataset is widely used as a benchmark dataset for evaluating the performance of intrusion detection systems (IDSs) and machine learning algorithms. It is an updated version of the original KDD Cup 1999 dataset, addressing some limitations and criticisms of the original dataset, such as the presence of redundant records and unrealistic traffic patterns.

ii. Realistic Network Traffic: The NSL-KDD dataset contains a representative sample of network traffic data, including both normal and various types of attacks, such as DoS, probing, and unauthorized access. This diversity allows researchers to study and develop IDSs and machine learning models that can effectively detect and classify different types of network intrusions.

iii. Feature Selection Challenges: The NSL-KDD dataset provides a significant number of features (41 features) that capture various aspects of network traffic. This feature-rich dataset allows researchers to tackle the challenge of feature selection and

extraction, exploring the most relevant features for intrusion detection and determining the optimal subset of features for achieving high detection accuracy.

iv. Pre-processed and Labeled: The NSL-KDD dataset is pre-processed and labeled, meaning that the raw network traffic data has been transformed into a suitable format for machine learning. Additionally, each network connection is labeled as either normal or belonging to a specific attack category, making it convenient for supervised learning tasks.

v. Availability and Reproducibility: The NSL-KDD dataset is readily available for download and has been used extensively in research publications. This availability enables researchers to compare their results with existing literature, reproduce experiments, and foster collaboration among the research community.

However, it's worth noting that the NSL-KDD dataset is not without limitations. Some criticisms include the fact that it may not fully represent the complexities and diversity of real-world network traffic, as it was generated in a controlled lab environment. Therefore, researchers should be cautious when generalizing results obtained from the NSL-KDD dataset to real-world scenarios. Overall, the NSL-KDD dataset serves as a valuable resource for exploring and developing machine learning-based intrusion detection systems and provides a standardized platform for evaluating the performance of different techniques in the field of network security.

Most of the approaches in machine learning required a training dataset to train the algorithm before being tested. Usage of the dataset in classifying attack has been done in [9] and a total of 63% of the reported research used the KDD-99 dataset for training and testing. In this work, a newer version of KDD-99 which is an NSL-KDD dataset will be used. The dataset has been corrected from enormous duplication of data and defects that cause bias in evaluation. NSL-KDD dataset has 41 features and three of these features are symbolic while others are numeric. Our model is trained by the KDDTrain+ dataset and tested by the KDDTest+ dataset. In addition, the KDDTest+ dataset contains 17 different types of attacks compared to 22 attack types of the KDDTrain+ dataset. Thus, the KDDTest+ dataset is a reliable indicator of the performance of the model on zero-day attacks as well. Distribution of both dataset according to network traffic classes are as shown in Table 3. 1 below.

KDDTrain+ dataset contains 125,973 records and KDDTest+ dataset contains 22,544 records. The data processing task will be deployed to prepare the dataset for training and testing purposes. Simplified autoencoder with the fine-tune of overfitting factors is applied to the NSL-KDD training and testing dataset and only selected 6 features from Section III. The selected features are service, flag, src bytes, dst bytes, logged in and serror rate.

Dataset	Normal	Dos	Probe	R2L	U2R	SUM
Train+	67,343	45,927	11,656	995	52	125,973
Test+	9,711	7,458	2,754	2,421	200	22,544

Table 3. 1: Distribution of KDDTrain+ and KDDTest+ dataset according to traffic classes

Many of the features of the NSL-KDD dataset have very large ranges between the maximum and minimum values, such as the difference between the

maximum and minimum values in “duration [0, 58329].” In this example, the maximum value is 58,329 and the minimum is 0. A large difference also exists in other feature values, such as “src-bytes” and “dst-bytes” thereby making the feature values incomparable and unsuitable for processing. Hence, these continuous features are normalized by using max-min normalization for mapping all feature values to the range [-1, 1] according to equation (3.1) below**.**

$$x_i = \frac{x_i - \mathrm{Min}}{\mathrm{Max} - \mathrm{Min}} \qquad \textbf{(3.1)}$$

Features that are discrete such as service and flag are being processed using a one-hot encoding. As the speed of processing is a concern, using a one-hot implementation typically allows a state machine to run at a faster clock rate than any other encoding of that state machine. Both of these values will then be used as input to the model for training and testing.

B. AERF Model

Deep learning approaches have good potential to achieve effective data representation for building improved solution. For the approach to be adopted in the real environment, it has to be composed with a fast simplified approach for performance. Therefore, in this proposal, an autoencoder with fine-tuning the factors that affect overfitting is being put forth. An autoencoder is an unsupervised machine learning algorithm that learns to encode and decode data. It consists of an encoder and a decoder network that work together to reduce the input data into a compressed representation, also known as a latent code. The encoder network takes the input data and maps it to the latent

code, while the decoder network takes the latent code and maps it back to the reconstructed data.

The objective of the autoencoder is to learn a compressed representation of the input data that can be used to reconstruct the original data with minimal loss of information. During training, the autoencoder is fed with input data and the corresponding reconstructed data, and the weights of the encoder and decoder networks are updated to minimize the difference between the input data and the reconstructed data. A hybrid combination using the random forest algorithm for further classification from the preliminary autoencoder calculation process is proposed. A preliminary test with the usage of the NSL-KDD dataset will be used in order to test the approach by targeting high accuracy detection but with the main objectives of simplifying the detection within the minimum time processing needed. The AE has two phases which are encoding and decoding. For the encoding process as in equation (3.2) input data x is compressed into a low-dimensional representation h and then the decoder reconstructs the input based on the low-dimensional representation.

$$\begin{aligned} \mathrm{h} &= \mathrm{f}(\mathrm{Wx} + \mathrm{b}) \\ \mathrm{y} &= \mathrm{f}(\mathrm{W}'\mathrm{h} + \mathrm{b}') \end{aligned} \tag{3.2}$$

Where $\mathrm{f}(\cdot)$ is a non-linearity activation function, W and W' are hidden weight matrices, b and b' are biases and y is output vector. Minimizing the differences between the input x and the output y is the main goal during the training process. The average squared difference between the estimated values and the actual values is used as below in equation (3.3).

$$L(x, y) = \left||x - y|\right|_2^2 \quad \textbf{(3.3)}$$

During the training process, only normal traffic is used from the training dataset. Attack datasets not used because of unbalanced and certain attack types are not well represented. In this approach, autoencoder with dropout on the inputs is proposed which consists of an input layer of 85 neurons due to the number of features for each sample is 85. The input layer of 85 neurons after the selection of 6 features is used for the training and testing process. 85 neurons are consists of service (70 input), flag (11 input), src bytes (1 input), dst bytes (1 input), logged in (1 input) and serror rate (1 input).

A total of 125,973 with 85 input and 22,544 with 85 input for dataset training and testing respectively. After a dropout layer, a hidden layer of 8 neuron units is proposed with the hidden representation of the autoencoder has a compression ratio of 85/8 forcing it to learn interesting patterns and relations between the features. An output layer of 85 units with the activation of both the hidden layer and the output layer using the most popular rectified linear unit (ReLu) activation function as shown in Figure 3. 3 below. It gives an output x if x is positive and 0 otherwise.

$$A(x) = \max(0, x) \quad \textbf{(3.4)}$$

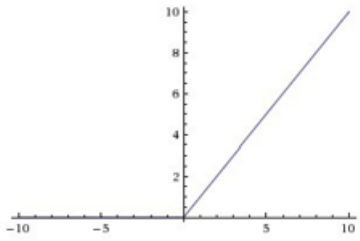


Figure 3. 3 : ReLu activation function

During the reconstruction of the inputs, the autoencoder was trained using only the samples labelled "Normal" in the training dataset. This is for the purpose of allowing it to capture the nature of normal traffic and to minimize the mean squared error between the input and output. Preventing the autoencoder from simply copying the input to the output and overfitting the data, regularization constraints are enforced. Model architecture details are as shown in Figure 3. 4 and the parameter used for the model depicted in Table 3. 2.

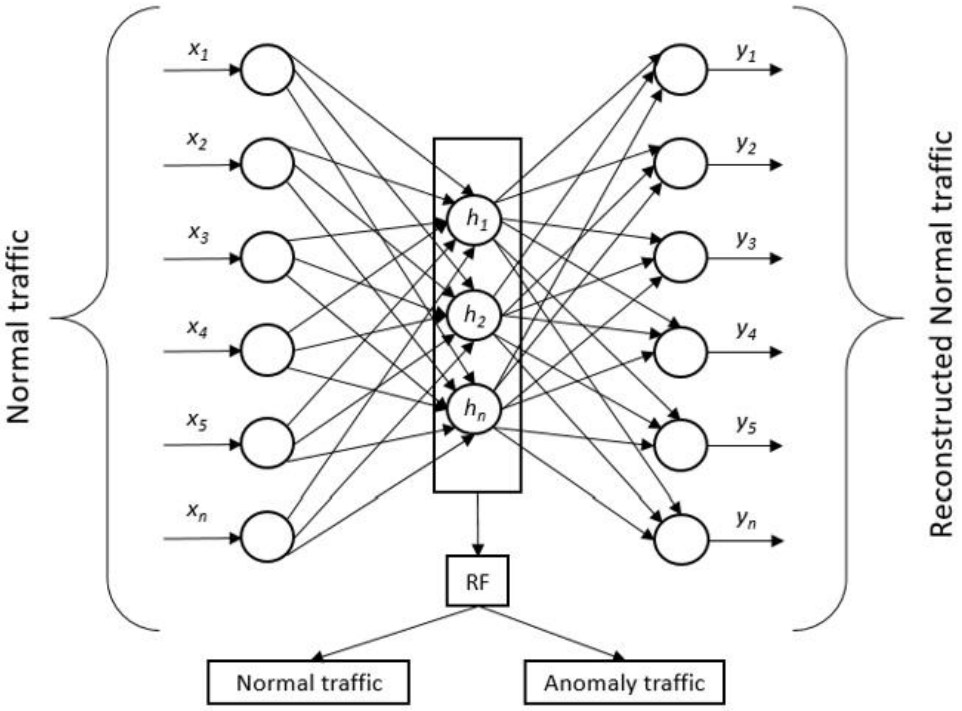


Figure 3. 4 : AERF Model

Input layer	Dropout	Hidden layer	Act regularizer	Output layer	Activation Func
85	0.5	8	10e-5	85	ReLu

Table 3. 2 : AERF model parameter

The hidden layer of the model is being used as the input for the second layer classification using random forest algorithm. Random forest applies the principle of ensemble learning method on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Examples of research works that employ RF for intrusion detection is such as [35] and [36]. Comparative studies with regards RF as one of the best algorithms selected for a similar area being concluded by the works of [10] and [11]. In our proposed model, the encoded representations learned by the autoencoder is being used as the input for the RF classifier to classify either normal or anomalous of the traffic.

3.4 Performance Evaluation

A. Evaluation Metrics

Performance evaluation metrics are used to assess the effectiveness and efficiency of various systems, models, algorithms, or processes. In the context of machine learning, there are several commonly used evaluation metrics depending on the specific task and goals. Performance evaluation metrics use in this evaluation is as below.

i. Accuracy: Accuracy is a widely used metric that measures the overall correctness of a classifier or model. It represents the proportion of correct predictions (true positives and true negatives) out of the total number of predictions.

ii. Precision: Precision measures the proportion of true positive predictions out of the total positive predictions made by a classifier. It

focuses on the correctness of positive predictions and is useful when the cost of false positives is high.

iii. Recall (Sensitivity or True Positive Rate): Recall calculates the proportion of true positive predictions out of the actual positive instances. It assesses the ability of a classifier to correctly identify positive instances and is important when the cost of false negatives is high.

iv. F Measure: The score is the harmonic mean of precision and recall. It provides a balanced measure of a classifier's performance by considering both precision and recall. It is useful when the dataset is imbalanced.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \textbf{(3.5)}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \textbf{(3.6)}$$

$$\text{F} - \text{measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad \textbf{(3.7)}$$

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad \textbf{(3.8)}$$

These metrics are calculated by using four different measures, true positive (TP), true negative (TN), false positive (FP) and false negative (FN):

TP: the number of anomaly records correctly classified.
TN: the number of normal records correctly classified.
FP: the number of normal records incorrectly classified.
FN: the number of anomaly record incorrectly classified

B. Experimental Results

The model is trained for 10 epochs using an Adam optimizer with a batch size of 100 as shown in Fig. 5. A total of 60,608 samples are used for training and 10% of the total normal traffic which are 6735 samples as validation. Tensorflow backend is used to run and simulate the proposed as in Figure 3. 5 below.

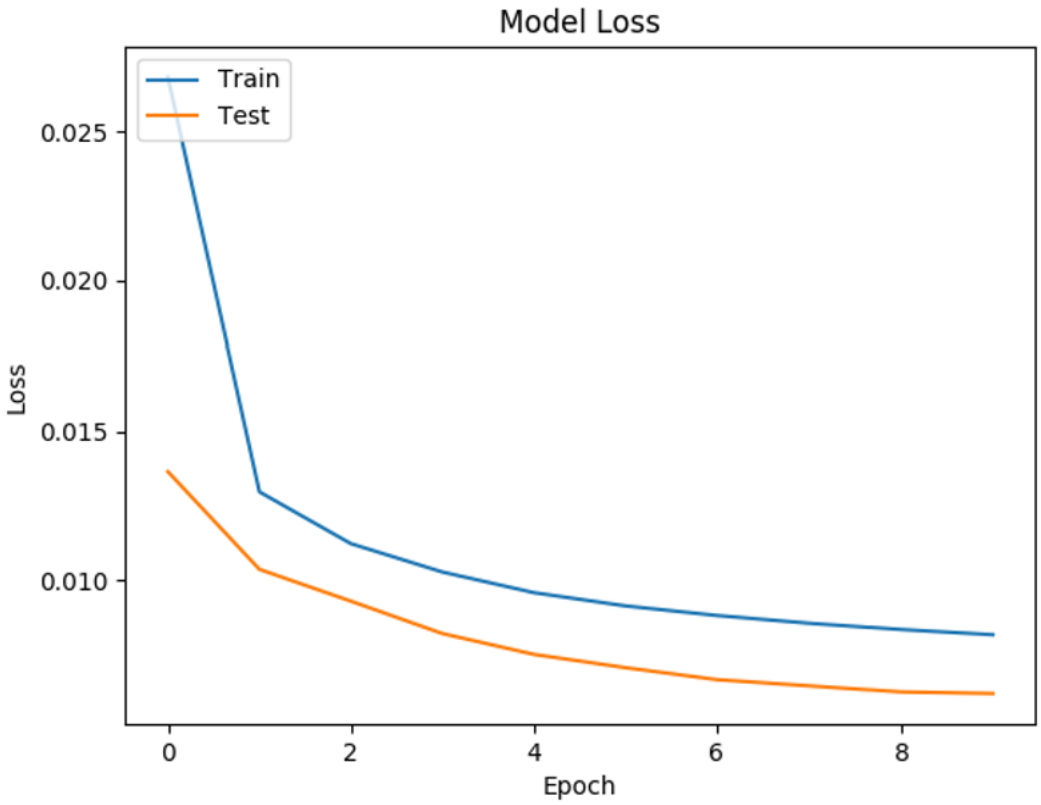


Figure 3. 5 : Model training process

During the training process of a machine learning model, the training loss is a metric that quantifies how well the model is performing on the training data. It represents the discrepancy between the predicted outputs of the model and the actual labels or targets in the training set. The specific calculation of the training loss depends on the type of machine learning task and the chosen algorithm. In supervised learning, where the model is trained to predict labels

or continuous values, the training loss is typically computed using a loss function, such as mean squared error (MSE) or cross-entropy loss. The training loss is minimized during the training process through an optimization algorithm, such as gradient descent or stochastic gradient descent. The goal is to iteratively update the model's parameters in a way that reduces the training loss and improves the model's ability to generalize to unseen data. Monitoring the training loss over successive epochs or iterations provides insights into the model's learning progress. A decreasing training loss indicates that the model is converging towards a better solution, while an increasing or stagnant loss may indicate issues such as underfitting or inadequate model capacity. It's important to note that while minimizing the training loss is a crucial aspect of model training, it is not the sole metric for evaluating model performance. It is essential to assess the model's performance on separate validation or test sets to ensure that it can generalize well to unseen data and avoid overfitting, where the model performs well on the training data but poorly on new data.

Repetition of training and test has been performed with stabilized outcome. Results that have been collected from the test are promising with the mean accuracy of 90.19% achieved as shown in Table 3. 3 from the autoencoder process alone.

Method	Accuracy	Precision	Recall	F1
Autoencoder	90.19	88.78	94.70	91.64

Table 3. 3 : Autoencoder performance over test dataset

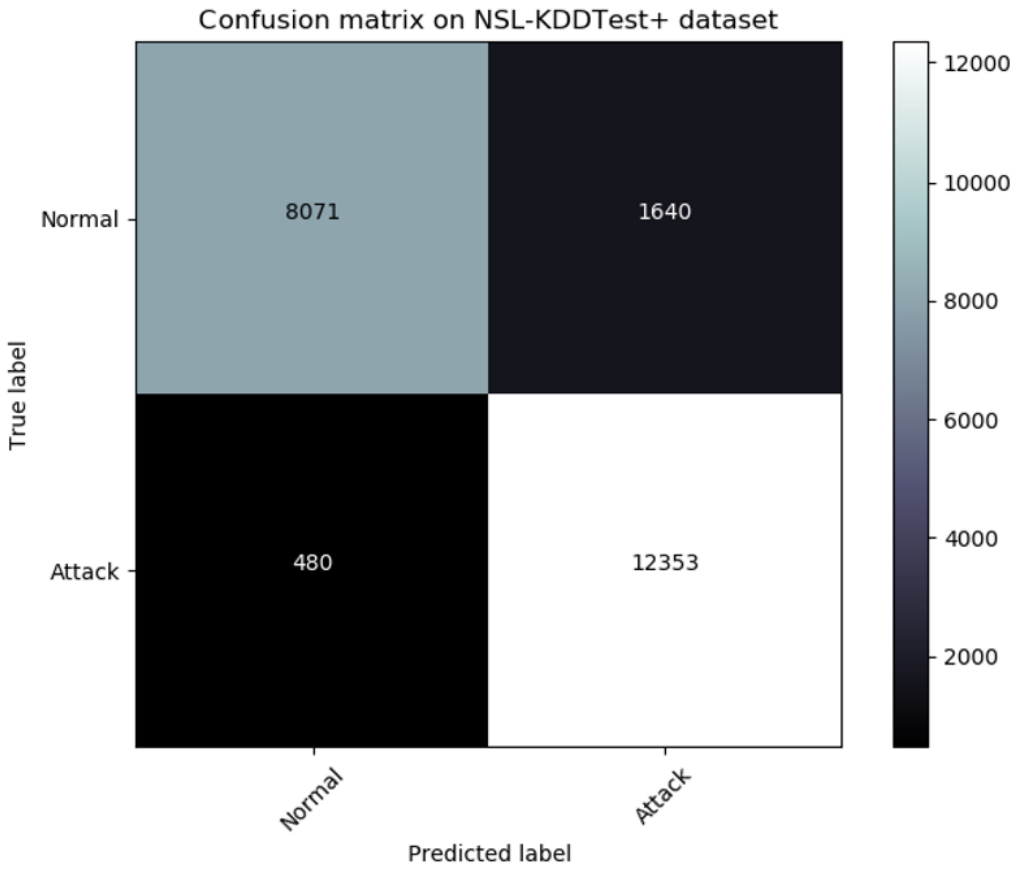


Figure 3. 6 : Autoencoder confusion metrics

Example confusion matrix from the autoencoder process for the test dataset is as shown in Figure 3. 6. The threshold for autoencoder is based on the calculated loss from the training process for the separation of normal and anomaly traffic. Distribution of detection from the autoencoder is as shown in Figure 3. 7 below. It is noticed that the error rate for normal and abnormal traffic is highly distinctive from the calculated threshold value. An average amount of 10% of the calculated error is in the wrongly determine area. As for the next process, the encoded representation of the autoencoder is inputted into RF for a further breakdown of the category of the traffic.

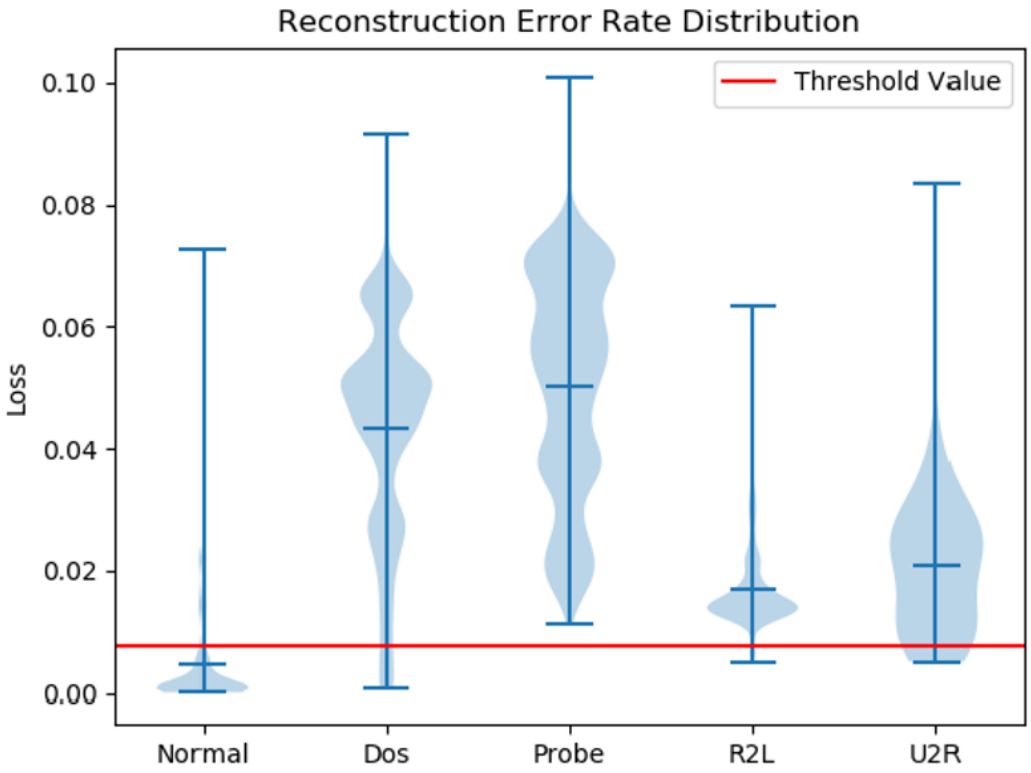


Figure 3. 7 : Distribution of detection over test dataset

RF classification that has been adopted has yielded an accuracy of 98.4% for the input from the autoencoder. Comparison from previous related works that have also using the same NSL-KDD dataset and intrusion detection are tabulated in Table 3. 4. An increase in accuracy is achieved with the proposed model and enables further improvement to be carried out.

Method	Accuracy Rate
Naive Bayesian [12]	75.56%
MLP [12]	77.41%
Random Forest [12]	80.67%
J48 [12]	81.07%
RNN [13]	83.28%
Fuzzy approach [14]	84.12%
AE + Gaussian NB [15]	84.34%
STL-IDS [12]	84.96%
S-NDAE [5]	85.42%
SAE+SMR [6]	88.39%
GRU-LSTM [4]	89.00%
AERF	**98.40%**

Table 3. 4 : Performance comparisons in binary classification using KDDTrain+ and KDDTest+ dataset

3.5 SDN Controller Performance

In this section, we evaluate the effect of the AERF on the SDN network performance. The evaluation testbed is described in the first part and then the network performance evaluation is presented.

Experimental Setup

The AERF is implemented as an application written in Python language in a Ryu controller. Cbench is a standard tool used for evaluating the SDN controller performance which supports two running modes, throughput and latency. The throughput mode computes the maximum number of packets handled by the controller and latency mode computes the time needed to process a single flow by the controller. We run our experiments on a virtual machine having an Intel(R) Xeon(R) E3-1226 3.3GHz with 3 cores available

and 8GB of RAM. The operating system is Ubuntu 18.04.2 LTS 64bit. The controller performance is tested with a different number of virtual OpenFlow switches emulated by Cbench. The performance of the stand-alone Ryu controller is considered as a baseline for our evaluation.

Analysis of Results

The average response rate of the controller for both the baseline and AERF evaluation is as depicts in Figure 3. 8. The throughput achieved a slight decrease when the number of switches increases. Although the throughput decreased, comparison with the baseline stand-alone Ryu controller is not so much different and still acceptable. An interesting observation is the Ryu controller which has a negligible impact on its latency performance as shown in Figure 3. 9. When we increase the number of connected switches, the latency achieved are between 3 to 4 ms and not so much overhead introduced. This is a similar commentary that has been done by [37] regarding the latency of the Ryu controller performance achieved. A comparison of time taken for training and testing NSL-KDD dataset are as shown in Table 3. 5 below.

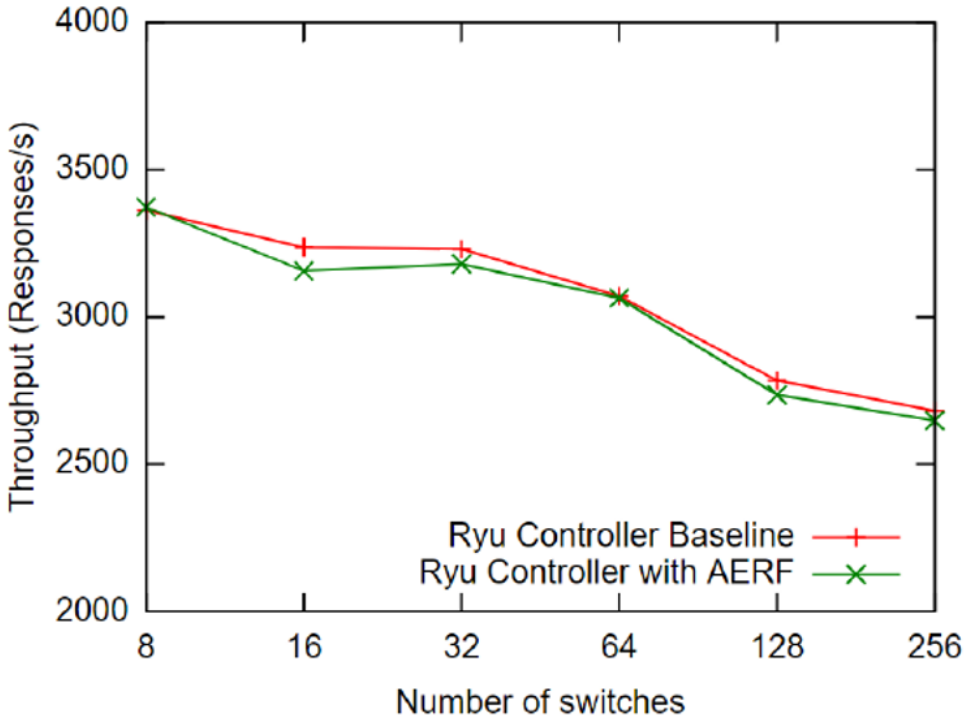


Figure 3. 8 : SDN environment throughput evaluation

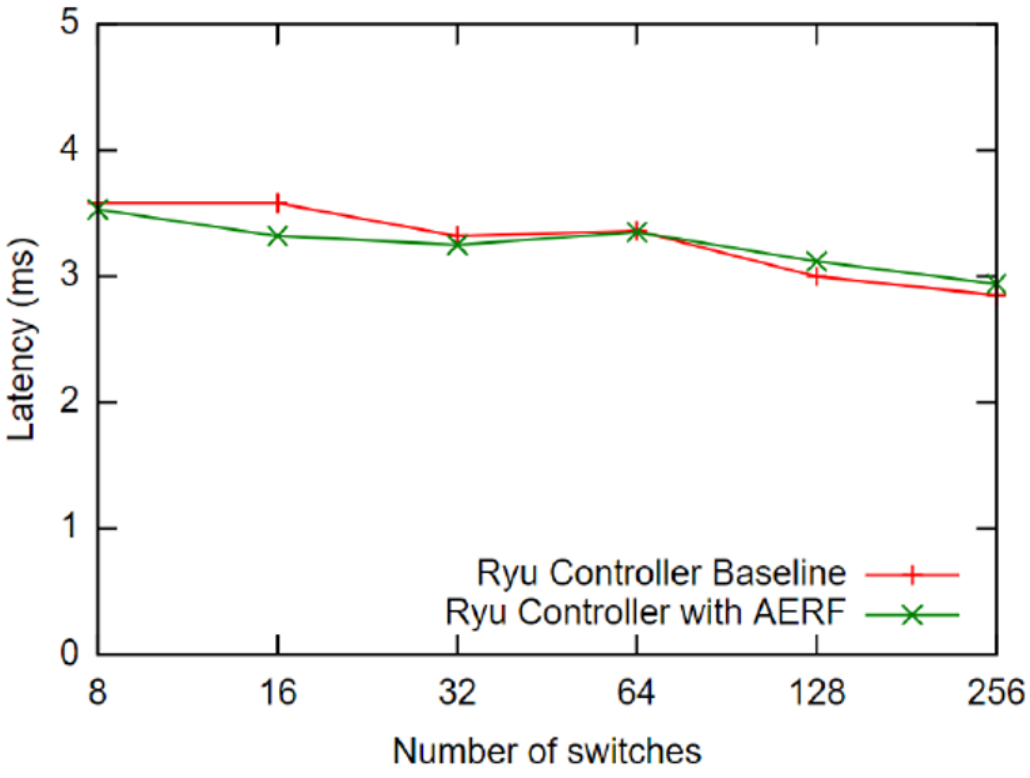


Figure 3. 9 : SDN environment latency evaluation

Method	Training Time	Testing Time
STL-IDS [12]	673.031	4.468
S-NDAE [5]	644.84	-
AERF	**18.32**	**2.63**

Table 3. 5 : Performance comparisons in binary classification using KDDTrain+ and KDDTest+ dataset

Simplicity is the focus of this approach. The simplified single hidden layer of minimum neurons has made the training and testing process very fast which are suitable for the SDN environment which demands high processing capability and real-time implementation. Adaptive calculation of threshold within the model has bypassed the needs of human intervention to manually set it. The value of the threshold could also be manually adjusted in order to suit the needs between sensitivity and specificity accordingly. The accuracy achieved with the very minimum processing time is a good sign of future applicability in real SDN environment deployment.

3.6 Summary

The chapter presents a novel approach called AERF (Autoencoder and Random Forest) for intrusion detection in a native SDN environment. The AERF method combines the power of an autoencoder, a type of neural network used for unsupervised learning and feature extraction, with the random forest algorithm, a popular supervised learning technique.

The results of the study demonstrate that the proposed AERF approach outperforms previous works in terms of accuracy, achieving an impressive 98.4% accuracy rate. Furthermore, the AERF model exhibits reduced training and execution time compared to alternative methods, making it efficient for real-time intrusion detection.

The works also conducted experiments to assess the impact of the proposed approach on the performance of the SDN controller. The results show that the AERF approach does not significantly affect the controller's throughput and latency. This finding indicates that the proposed method is practical for implementation and can be adapted within an SDN-based intrusion mitigation architecture for further prevention processes.

Overall, the chapter highlights the efficacy of the AERF approach for intrusion detection in an SDN environment, showcasing its high accuracy, reduced training and execution time, and minimal impact on SDN controller performance. The promising results suggest that the proposed method holds potential for real-world deployment and can contribute to enhancing the security of SDN networks.

CHAPTER 4 : Hybrid Deep Autoencoder with Random Forest in Native SDN intrusion detection Environment

4.1 Introduction

The advantages and features of the SDN environment have been widely studied to help solve new network security challenges. Deploying intrusion detection systems (IDS) as a method of combatting these threats, is often proposed. In general, IDS are grouped into two types. The first depends on the pre-identified signature created from a series of known intrusions recorded previously. The second type monitors traffic behaviour to capture abnormalities from regular traffic deemed suspicious. Machine learning features and capabilities have been adapted to provide an efficient system with a high level of security in various areas such as authentication, access control, malicious outbreak detection, and others [32]. A subset of machine learning domains is deep learning. Deep learning algorithms allow a given dataset's features to be extracted systematically to create sequential layer architecture, which is then applied using a non-linear transformation function to build up the base of deep learning algorithms. The complexity of the non-linear transformation constructed increases in parallel with the number of layers being used. The algorithm learns the abstract, hidden properties of the data obtained from the last layer, representing the multiple levels of abstract representation acquired during the process. Finally, introducing the data into a high-level non-linear function results in the abstract properties being gained. Integrating machine learning and deep learning into research relating to enhancing network security has been carried out together with the advances and increased capabilities offered by SDN architecture [38]. A subset of deep

learning, an autoencoder is unsupervised learning technique for undertaking representation learning using a neural network [39]. The workings of [9] and [34] have used the autoencoder technique for detecting intrusion in the network. The method creates a bottleneck to capture compressed knowledge from the original data. An optimum model can be achieved by manipulating the sensitivity of the input for a precise reconstruction of build-up, taking into consideration the overfitting condition resulting from copying the input data. To handle sensitivity, a loss function is commonly used to minimize loss while discouraging any overfitting or memorizing of input data, so a parameter such as a regularizer is added to the model. Excellent results from adopting an autoencoder with machine learning capabilities offer opportunities to create similar solutions, especially in the intrusion detection domain in the SDN environment. A further exploration aiming to increase accuracy levels, minimize processing time, and other benefits can still be added as an improvement. This possibility motivated this research to contribute to further improving the native SDN environment. To summarize, the contributions of this chapter are:

- The introduction of a hybrid combination of a deep autoencoder (DAE) and random forest (RF) algorithm in the native SDN environment, deployed in the SDN controller mechanism.
- Within the SDN Controller, our DAERF approach yields a detection rate of 98% using purely native SDN statistics collection as features.
- We also compared the performance achieved with the original dataset and previous research to investigate the developed model's performance. The results show that our approach has significant potential for real-time detection when deployed in the SDN controller.

4.2 Architecture

In building a typical machine learning model, essential input features are selected, and the model automatically learns by mapping identified characteristics of features to a conclusion output. In deep autoencoder, there are multiple levels of encoding, and decoding used. Abstract features from various levels are automatically being discovered and composed to produce output. The features from the previous level are carried forward to the next level to be processed again for another level of abstract representation. Our experiment constructed a deep autoencoder model with an input layer, three hidden layers, and an output layer. A total of 8 dimensions for input and output are selected in the autoencoder. The hidden layers contain six, four, and two neurons, respectively. The middle, hidden layer of two neurons is used to input the random forest classifier for the intrusion detection process. A total batch size of 1000 were trained for ten epochs using the chosen Adam optimizer. Two phases are involved in the autoencoder, namely encoding and decoding. Input data x is compressed into a low dimensional representation h during the encoding process, while the low-dimensional representation is reconstructed into input data during the decoding process. A non-linearity activation function represents with f (·), hidden weight matrices indicated by W and W', biases denoted by b and b'', and y is output vectors. The main goal of training the DAE is to minimize the difference between the input x and output y. Therefore, an MSE loss function is used as shown in (4.1) below.

$$L(x, y) = ||x - y||_2^2 \quad \textbf{(4.1)}$$

In this approach, a stacked autoencoder with dropout on the inputs is proposed, consisting of an input layer of 8 neurons due to the number of features for each sample is 8. After the initial dropout process, a combination of 6,4 and 2 neuron units within the stacked hidden layer is proposed. The autoencoder uses an 8/2 compression ratio to allow it to learn relations between the chosen features and exciting patterns that might be discovered. Rectified linear unit (ReLu) activation function was used for the eight units of the hidden layer and the output layer. It gives an output x if x is positive and 0 otherwise. For optimization steps, regularization constraints are enforced to avoid the autoencoder overfitting the data by copying the input directly to the output. The architecture of the model is shown in Figure 4. 1 and Table 4. 1 presents the model's parameters. Random Forest was tested and selected as the classifier with the best detection performance and lowest time usage [40], [41], [36], [42] and [43].

Input layer	Output layer	Hidden layer	Dropout	Act regulizer	Act Func
8	8	6,4,2	0.5	10e-5	ReLu

Table 4. 1 : Model parameter

For this, it was used to process the model's hidden layer output in the second layer classification. The output becomes the input for the classification algorithm. The principle of random forest uses the ensemble learning method on the dataset sub-sample by averaging to control for over-fitting and thus improve prediction accuracy. Examples of research that employs random forest classification for intrusion detection include [35] and [36]. The work of [44] and [45] also concluded that random forest is among the top algorithms for similar intrusion detection in their comparative studies. Encoded

representation from the autoencoder is passed to the random forest algorithm for the final classification process. The traffic is then categorized as normal or an attack attempt, which the SDN controller will penalize further through traffic rule action.

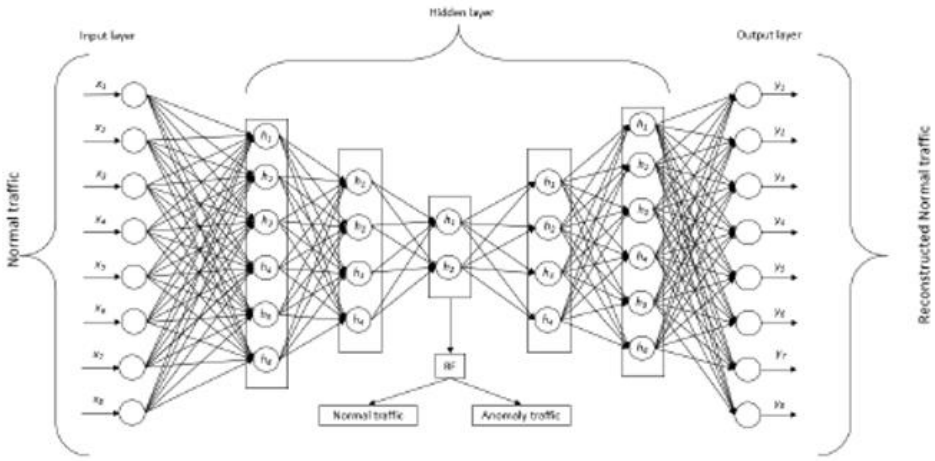


Figure 4. 1 : Deep autoencoder with random forest model

4.3 Experimental Methodology

Machine learning methods require a dataset to be well built up. For such a dataset, a training dataset for the developed algorithm is needed before the algorithm is tested. KDD-99 dataset was the most popular dataset used in this field, and 63% of research has adopted this dataset. In this work, we focus on the deployment part of statistics collection in a native SDN environment. In

order to collect actual SDN traffic activities, a simulation of real network activities needed to be made. CICIDS2017 dataset provides the total payload packets in PCAP format, replicating actual network traffic activities in the targeted environment. This dataset covers seven types of common attack

groups (i.e., Botnet, Brute Force Attack, DoS Attack, DDoS Attack, Heartbleed, Infiltration Attack, and Web Attack). This dataset consists of a Microsoft Excel CSV dataset and complete payload packets in a PCAP format file for a five day working hours period. The PCAP files are each sized7 to 12 GB . Each flow sample of the dataset contains 83 flow features. For this, a total of 16 native SDN OpenFlow flow features and a binary classification type were collected during the PCAP traffic emulation in a simulated SDN environment. Out of the 16 collected features, eight flow features have been selected for further machine learning analysis for intrusion detection in a native SDN environment. Another eight features have been dropped because the information differs for each network connectivity and does not represent the analysed traffic pattern. After the PCAP file had been injected into the SDN environment, there were a total of 263156 collected flows for Monday, 227271 collected flows for Wednesday, and 230290 collected flows for Friday. A total of 576,573 samples were used for training and 20% of the total normal traffic, 144,144 samples for testing. Mininet [46] was used as a network emulator with the Ryu [47] component-based SDN as the framework. The proposed model simulation used Tensor flow [48] backend . The model's training and testing performance are shown in Figure 4. 2 below.

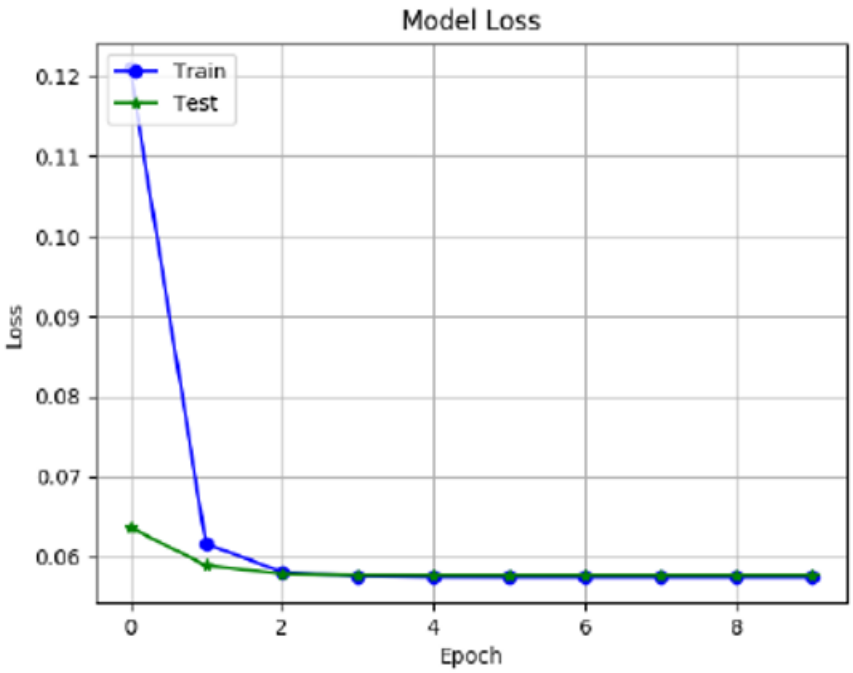


Figure 4. 2 : Training and testing of the model

A standard measure for classifier comparison is shown by the Receiver Operating Characteristic (ROC) curve. Plotting the false positive rate versus the true positive rate produces the curve for the ROC. The area under the curve (AUC) is to determine the model's performance in predicting the classes. The higher the AUC, the better the classifier, as shows, the proposed model achieved 0.92 AUC.

4.4 Performance Evaluation

4.4.1 Evaluation Metrics

Four (4) types of metrics were used for the evaluation: Accuracy, Precision, Recall, and F-measure.

True Positive, True Negative, False Positive, and False Negative are measured as follows:

True Positive: the number of anomaly records correctly classified.

True Negative: the number of normal records correctly classified.

False Positive: the number of normal records incorrectly classified.

False Negative: the number of anomaly records incorrectly classified

4.4.2 Experimental Results

The reiteration of training and testing in the experiment resulted in a persistent outcome. The mean accuracy achieved was 99% for training and 98% for testing. Table 4.2 provides a comparison of the achieved performance with other related research studies.

The performance of the proposed approach was found to be better than the results presented in [40]. However, it did not surpass the performance of the two other research works mentioned in [49] and [50]. It should be noted that the scope of the experiments in these studies differed. [49] utilized tshark for network traffic dumping and analysis, while [50] employed the CICIDS2017 dataset in Weka. In contrast, the current research was conducted purely within an SDN controller using native OpenFlow communication traffic.

Despite the slightly lower performance compared to the referenced studies, the researchers acknowledge the potential for improvement in their approach within the native SDN environment. Future efforts will be focused on refining the methodology to enhance the achieved results.

In summary, the experiment consistently yielded high mean accuracy values of 99% for training and 98% for testing. While the performance was not as strong as certain other studies, the unique implementation in the native SDN environment provides valuable insights and sets the stage for further advancements in the proposed approach.

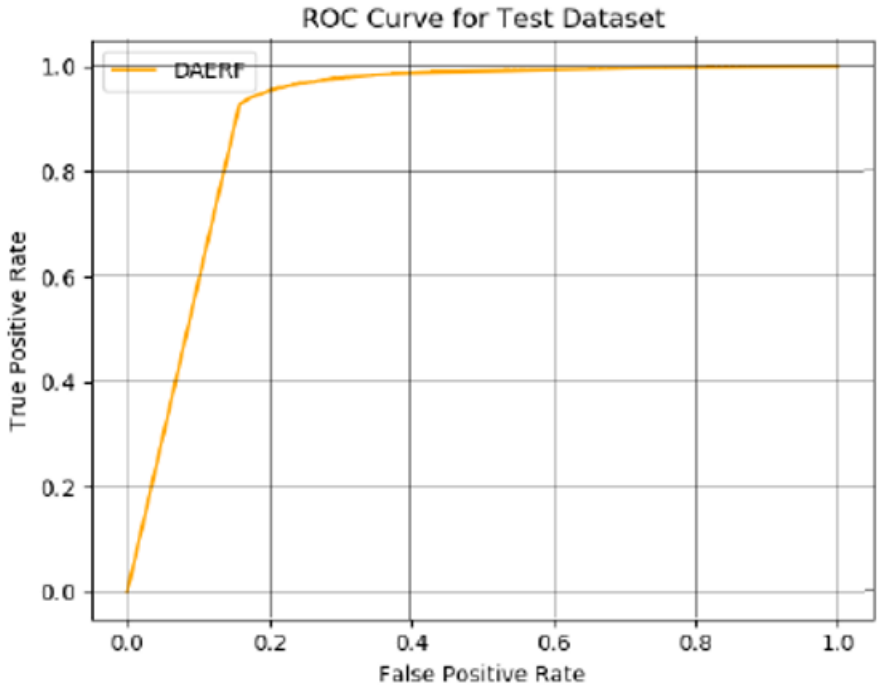


Figure 4. 3 : ROC AUC curve

Research	Accuracy	Precision	Recall	F1
[6]	99.9	99.8	99.9	99.8
[7]	99.8	99.9	99.8	-
[13] Adaboost	-	77	84	77
[13] ID3	-	98	98	98
[13] KNN	-	96	96	96
[13] MLP	-	77	83	76
[13] Naive-Bayes	-	88	0.4	0.4
[13] QDA	-	97	88	92
[13] RF	-	98	97	97
Proposed solution	98	98.89	98.96	98.92

Table 4. 2 : DAERF performance over previous research

The same methodology explained above was tested using the CICIDS2017 excel files dataset. A similar experimental approach was developed in the SDN environment data collection, the same period of XLS dataset used. Excel files from the Monday, Wednesday, and Friday datasets were combined to create one new excel dataset. The dataset contains 83 flow features, and for the comparison test, eight features were selected from the excel files. The eight selected features represent similar characteristics to the features chosen in the SDN training. Table 4. 3 below, indicates the distribution of the combined datasets according to the network traffic classes. Data processing tasks such as normalization are also deployed to the dataset and prepare the dataset for training and testing purposes. The re-scale values are used as the input to the model for training and testing. The dataset was run into the model, and the performance captured with the same evaluation metrics.

Dataset	Attack	Normal	Total
Monday.csv	0	529,481	529,481
Wednesday.csv	251,723	439,683	691,406
Friday.csv	288,785	413,933	702,718
TOTAL	**540,508**	**1,383,097**	**1,923,605**

Table 4. 3 : Excel dataset distributions

A total of 1,538,884 samples were used for training, and 20%, 384,721 samples, for testing. The model evaluation process is shown in Figure 4. 4. The ROC AUC for the test dataset is shown in Figure 4. 5 below.

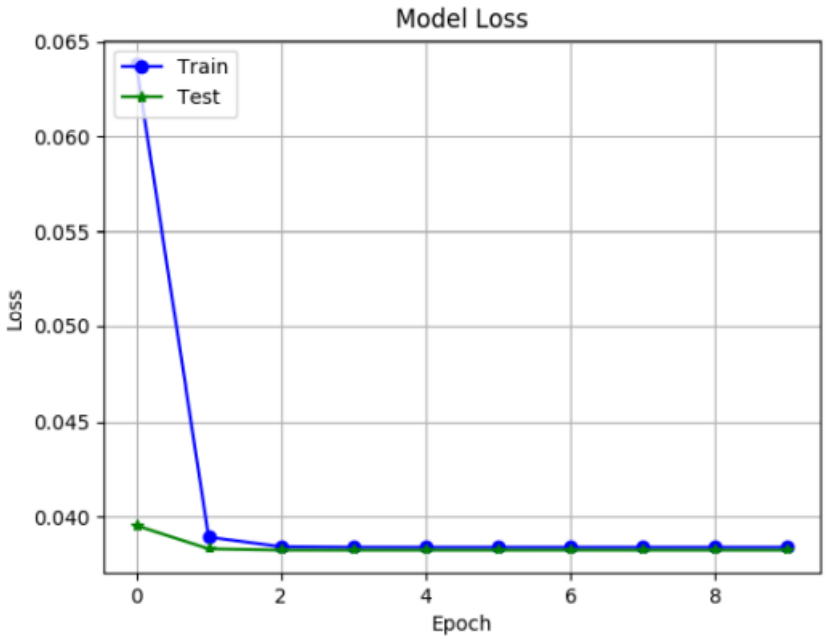


Figure 4. 4 : Model training and testing

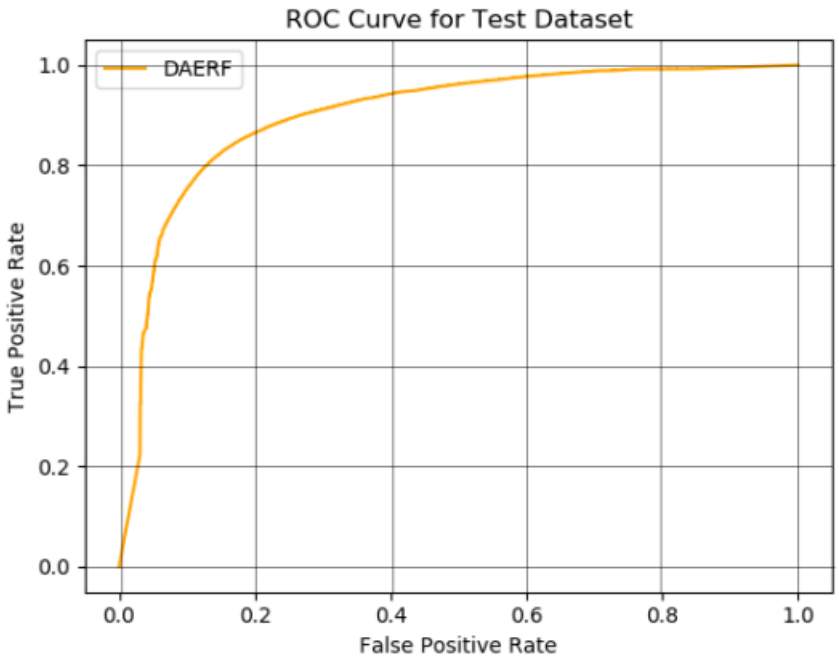


Figure 4. 5 : ROC AUC for test dataset

4.5 Summary

In this chapter, a hybrid model combining a deep autoencoder with a random forest classifier, referred to as DAERF, was introduced to enhance intrusion detection performance in a native SDN environment. The proposed model, implemented online in the SDN environment, demonstrated good performance with an accuracy and precision of 98%.

However, there are certain limitations to consider in this study. The PCAPs (Packet Capture) used in the experiments were sourced from the CICIDS2017 dataset, which contains both benign traffic and common up-to-date attacks. Although these PCAPs simulate real-world data in the context of the SDN environment, utilizing a full-scale real-world SDN environment would provide a more robust validation of the model's performance potential.

Nevertheless, the results obtained from the experiments indicate that the proposed model is an efficient tool for real-time intrusion detection in the SDN environment. Additionally, when compared with public intrusion datasets, the online implementation in the native SDN environment demonstrated better performance.

The chapter concludes by stating that the adoption of the proposed model in an SDN-based intrusion detection and mitigation architecture is both feasible and practical for further research. This suggests that the model has potential for real-world applications and can contribute to the advancement of intrusion detection systems in the SDN domain.

CHAPTER 5 : Adaptive framework for attack mitigation in SDN environment

5.1 Introduction

The annual Internet Report published by Cisco in 2020 predicted that the number of devices connected to IP networks will increase from 18.4 billion in 2018, to almost 30 billion by 2023. [51]. This enormous increase in the number of connected devices will attract a similar rise in the number of domain attacks or intrusions. The traditional way of handling attacks using conventional network device operations would face a hard task. Without global knowledge of the entire operating network and the ability to see every linked device's connections and behaviour, any defensive action would be less efficient, and handling invasion less effective. Software defined network (SDN) is an emerging technology that provides global knowledge and visibility through the separation of the control plane and data plane in the operational network. The control plane handles the centralized knowledge and the data plane provides the detailed activity collected from the transaction of the data in the network. Huge benefits are anticipated from an SDN approach with the help of research and experiments in intrusion detection. A lot of research has been done in order identify and prevent the invasion referred to above. Some work has focused on early detection, such as counting the number of connections, the entropy of transactions, and others [52], [53], [54], [21]. Embarking on machine learning capabilities is also being explored with the aim of studying the data representation and explicit meanings [7], [55]. Prevention steps and action are also being experimented with, aiming to minimize the impact of, and if possible, repel any attacks [56], [57], [58]. With

the focus of integrating and combining the different research areas into a complete approach, this chapter proposes an adaptive framework for attack mitigation in an SDN environment. To summarize, the contributions of this chapter are:

- The introduction of an adaptive framework for attack mitigation in SDN environments which present 3 layers of protection in an SDN architecture.
- Within the framework, we propose a three-layer protection mechanism for detecting and preventing attacks. It is entropy-based detection, hybrid machine learning in the control layer and proactive services monitoring in the application layer.
- We also compared the performance achieved with previous research to investigate the model's performance. The results show that our approach has great potential for adaptive, simplified detection and attack mitigation when deployed in the SDN environment.

5.2 Architecture

The proposed framework is shown in Figure 5. 1. The consolidated solution for the whole framework covers the layers within an SDN environment. Labelled number 1 , an entropy based module is the first layer of protection against an attack attempt. Traffic that has passed through the first layer is examined by the second module, which has hybrid machine learning intrusion detection located in the controller, as labelled in 2 . To mitigate the impact of an attack, the SDN controller monitors the services status of the public server being accessed by the traffic. Upon detecting a heavy load on the services

,the controller checks both earlier layer detections to handle the load coming in. This part of the observation is deployed in the application layer, as shown in 3 of the proposed framework.

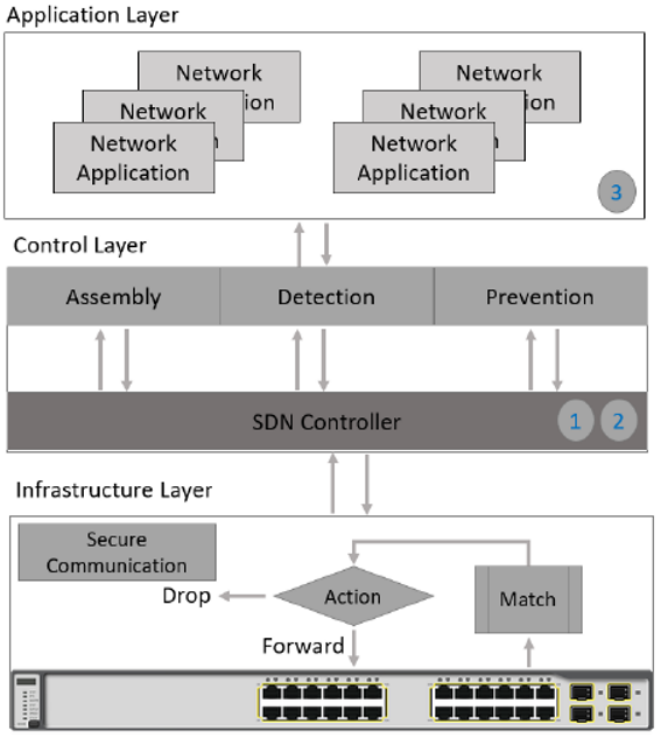


Figure 5. 1 : Application layer traffic monitoring

5.3 Experimental Methodology

A. Entropy based detection

Low-rate DDoS attack detection on the SDN controller is one of the most dangerous security concerns in SDN. The difficulty of identifying the assault comes from the attack traffic's similarity to typical traffic behaviour. When numerous hosts are involved, achieving high accuracy levels and a low false-positive rate becomes considerably more difficult. Meanwhile,

any detection technique must contend with high-rate DDoS attacks, especially when several targets are involved. As a result, the suggested technique uses passive monitoring of UDP packets in the SDN network to identify DDoS assaults on the SDN controller, independent of attack traffic and number of targets. A general R´enyi joint entropy is suggested in this study, based on integrating two concepts: the joint entropy approach and the R´enyi method. The general R´enyi joint entropy measures two random variables in the form of two packet header characteristics: the source IP address and destination IP address, which are represented by x and y, respectively, in the general R´enyi joint entropy. The R´enyi joint entropy approach is a combinations of joint entropy and renyi entropy, as shown in the equation below,

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{5.1}$$

$$H(X,Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log p(x,y) \tag{5.2}$$

$$H_{RJ\alpha}(x) = \frac{1}{1-\alpha} \log_2 \left(-\sum_{i=1}^{N} \sum_{j=1}^{M} p(x_i y_j)^{\alpha} \right) \tag{5.3}$$

Where $\mathrm{H_{RJ\alpha}(x)}$ is a R´enyi joint entropy, $\mathrm{p(x_i y_j)}$ is the probability distribution between source IP (x) and destination (y) during time interval (t) and signify a positive parameter. The R´enyi joint entropy technique is dependent on a value that can increase the detection rate by assessing the likelihood of traffic packets arriving. Based on the IP frequencies, the probability distribution $p(x_i y_j)$ is derived for each source

and destination. When each packet's probability distribution is evenly spread throughout all the hosts' destinations, a R´enyi joint entropy reaches its maximum value. The amount of probability assigned to all packets during a certain time window skewed towards a specific destination host produce a minimal value of R´enyi joint entropy. The R´enyi joint entropy is calculated using the likelihood each source IP address (x_i) and destination IP address (y_j) were recorded in the previous step within a certain timeframe.

B. Detection using hybrid ML

After the traffic has pass the entropy inspection, the next checking is done in the control plane layer and detection made using hybrid machine learning capabilities. A hybrid combination of autoencoder and Random Forest technique is deployed at this point, with detection focusing on attack that was not a type of DoS or DDoS attack. In the deep autoencoder, multiple levels of encoding and decoding are used. Abstract features from various levels are automatically discovered and composed to produce output. The features from the previous level are carried forward to the next level to be processed again for another level of abstract representation. Our experiment constructed a deep autoencoder model with an input layer, three hidden layers, and an output layer. A total of 8 dimensions of input and output are selected in the autoencoder. The hidden layers contain six, four, and two neurons, respectively. The middle, hidden layer of two neurons is used to input the random forest classifier for the intrusion detection process. In this

approach, a stacked autoencoder with dropout on the inputs is proposed, consisting of an input layer of 8 neurons, as the number of features for each sample is 8. After the initial dropout process, a combination of 6,4 and 2 neuron units within the stacked hidden layer is proposed. The autoencoder uses an 8/2 compression ratio to allow it to learn the relationship between the chosen features and any interesting patterns that might be discovered. Rectified linear unit (ReLu) activation function was used for the eight units of the hidden layer and the output layer.

C. Passive Application Layer Monitoring

A Ryu framework was adopted in our SDN topology .The controller plays two roles here: one as a standard SDN controller that controls and monitors the network, and the other as a defence mechanism. For the defence function, the controller keeps checking the status of the services being provided by the internal servers. A congested response from the servers via the status response indicates a high traffic flow and processing being handled by the targeted servers. In order to increase the response time for the server’s services, heavily connected connections will be penalized to provide the affected server with ample processing time to recover from the impact of an attack. As shown in Figure 5. 2, we utilized Monit application for monitoring the status of the server for detection of poor response of provided services.

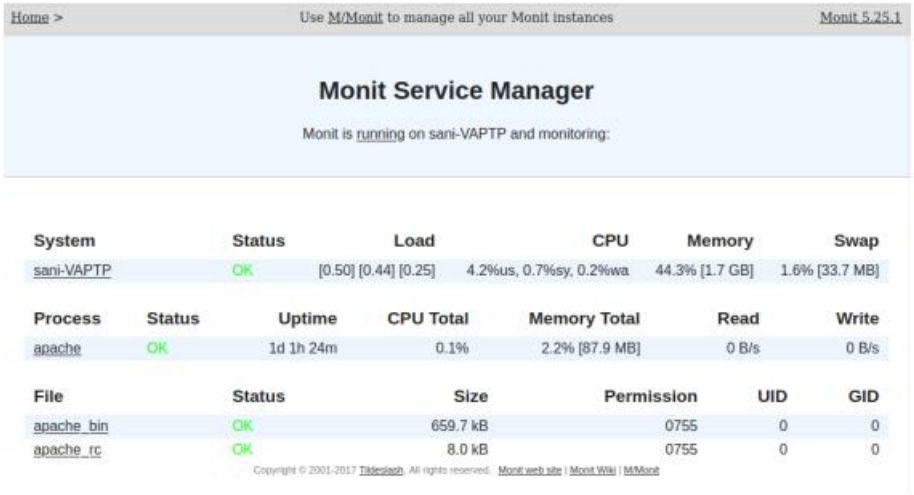


Figure 5. 2 : Passive monitoring using monit application

D. Dataset

The earliest balanced CICIDS2017 dataset, which includes millions of samples, is frequently utilized for product-level machine learning research. This dataset covers seven types of common attack (i.e. Botnet, Brute Force Attack, DoS Attack, DDoS Attack, Heartbleed, Infiltration Attack, and Web Attack). This dataset consists of complete payload packets in a PCAP format file and Microsoft Excel CSV dataset for a five day work period. Each flow sample in the dataset contains 80 flow features, explicitly explained in [23]. For this job, a combination of Wednesday and Friday afternoon with a DDoS attack dataset were chosen. Both days' dataset activities, which contain benign and also DoS/DDoS attacks were recorded. The selected portion from the overall CIDIDS2017, with sample size and class composition, are as shown in Table 5. 1 below.

Traffic Class	Label	Sample	Composition
BENIGN	BENIGN	537,749	58.550%
DDoS	DDoS	128,027	13.940%
Dos	DoS GoldenEye	10,293	1.120%
Dos	DoS Hulk	231,073	25.160%
Dos	DoS Slowhttptest	5,499	0.598%
Dos	DoS Slowloris	5,796	0.631%
Heartbleed	Heartbleed	11	0.001%
TOTAL	N/A	918,448	100%

Table 5. 1 : Traffic class distribution

5.4 Performance Evaluation

5.4.1 Performance Metrics

Evaluation Metrics

Two (2) types of metrics were used for the evaluation:

$$\text{AverageDetectionRate} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}} \quad \textbf{(5.4)}$$

$$\text{FalsePositiveRate} = \frac{\text{FalsePositive}}{\text{FalsePositive} + \text{TrueNegative}} \quad \textbf{(5.5)}$$

5.4.2 Experimental Results

We ran the CICIDS2017 Wednesday and a Friday dataset, which contains benign and DoS attacks, into the simulation as shown in Figure 5. 3 below. The controller's ability to detect DoS and DDoS attacks were tested during the entropy implementation. A window of 10-seconds was used to make the entropy calculation in order to determine whether traffic was benign or attacks. A total of 10 simulation runs were completed and the average rate for all 10

runs reported. The average detection rate was 98.16% and the false positive rate was 1.85%. The recorded detection rate and also false positive rate for the all 10 simulations are as shown Figure 5. 4 below.

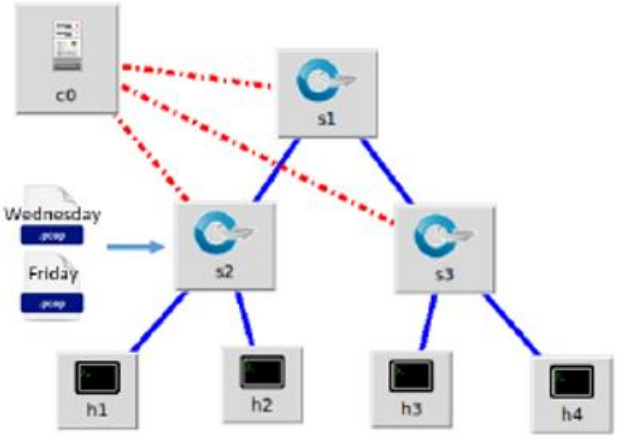


Figure 5. 3 : Pcap data injection

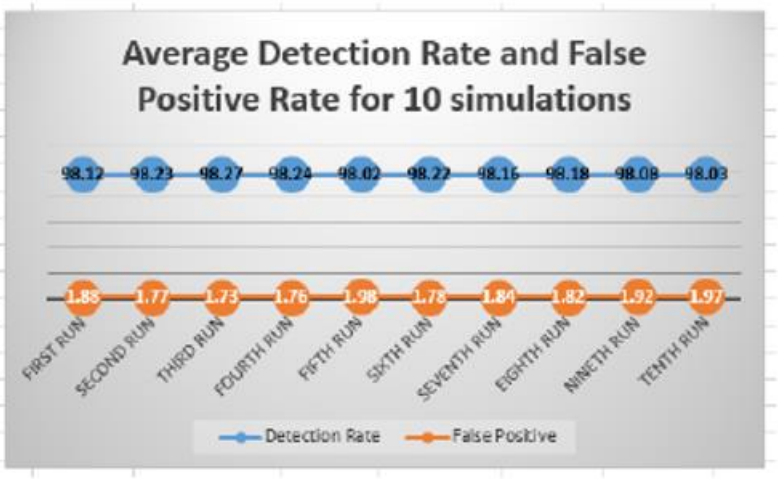


Figure 5. 4 : Average detection rate and false positive rate for 10 simulations

Table 5. 2 below, shows a comparison between the proposed solution and other, recent machine learning developments for DDoS attack detection in

SDN. For the purpose of comparison, the average detection rate and average false positive rate was used. In comparison to previous research for DDoS attack detection in SDN, the suggested technique delivers better results.

Research	Detection Rate	False Positive Rate
[24]	93%	9%
[24]	94%	6%
[25]	95%	5%
Proposed solution	98.16%	1.85%

Table 5. 2 : Performance comparison over previous researches

5.5 Summary

The solution presented in the chapter has both advantages and disadvantages. One of the disadvantages is that the separation of the control plane and data plane in the SDN environment can make the network infrastructure more vulnerable to cyber-attacks, particularly Distributed Denial of Service (DDoS) attacks. DDoS attacks pose a significant threat to SDN networks, as a successful attack can compromise the data and resources of the entire network.

To address this vulnerability, the chapter proposes a method for detecting DDoS attacks by maximizing the processing of traffic flow statistics. By simulating DDoS attacks and conducting experiments, the chapter demonstrates that combining multiple attack detection techniques within a simplified framework can leverage the full potential of SDN capabilities.

The experimental results suggest that this approach enables effective detection of both low-rate and high-rate DDoS attacks. Furthermore, by

combining multiple attack detection and mitigation methods, any weaknesses in one method can be covered by additional suggested solutions.

By adopting a comprehensive approach to attack detection and mitigation, the chapter broadens the possibilities for identifying and mitigating DDoS attacks in SDN networks. This approach has the potential to enhance the overall security and resilience of SDN environments.

However, it is important to note that while the proposed method shows promise, it is essential to continuously evaluate and update the detection and mitigation techniques to keep pace with evolving cyber threats.

CHAPTER 6 : Conclusion and Future Work

6.1 Conclusion

SDN are based on network programmability with a separation between control and traffic handling. The evolution of new architecture comes with new threats. Potential security loopholes emerge, creating new areas where protection is needed. Although it may seem new vulnerabilities only add to security concerns, nevertheless, finding countermeasures to solve these problems has increased researcher interest and motivation. The progress of new experiments into strengthening the SDN architecture have resulted in it benefiting from the use of SDN. For example, this thesis implemented an adaptive framework for attack mitigation in an SDN environment. A hybrid machine learning detection system can be deployed within SDN architecture to benefit the global overview of the network's intrusion detection. In Chapter 1, the limitations and constraints of the research were elaborated. The background to SDN architecture was discussed in Chapter 2, which includes how attack are implemented and their impact analysed. The various types of machine learning attack were also compared. Chapter 3 explored the capabilities of native SDN intrusion detection using machine learning, with a combination of autoencoder and Random Forest algorithm being deployed in a native SDN environment. The accuracy rate achieved was excellent, with a very small amount of time needed for testing the process. The model's improvement with the adoption of a hybrid deep autoencoder and Random Forest algorithms was described in chapter 4. The main difference between chapters 3 and 4 was the real recorded traffic from pcap files from the

CICIDS2017 dataset described in chapter 4. The recorded traffic was injected into the SDN environment to simulate real traffic behaviour, both normal and anomalous activities. Chapter 5 summarized the whole framework of an adaptive intrusion detection in SDN environments by incorporating 3 layers of intrusion detection in the architecture. Entropy-based calculations, detection by hybrid ML and passive application layer monitoring was proposed. This framework was tested and offered excellent detection rates and false positives rates . The final chapter, 6, summarized the whole thesis, with final remarks about the work that was done. There follows a brief summary of the key contributions this thesis makes.

- An adaptive framework for the SDN environment that can collect the important network parameters and monitor the whole network for intrusion detection was put forward. The data collected can then be processed to detect abnormalities in traffic transactions which it can then respond to and mitigate promptly.
- A hybrid ML was developed with a combination of mixed approaches to assess the traffic status through flow-based anomaly detection. The hybrid approach was shown to function with minimum impact on the overall SDN architecture.
- Recorded real-world traffic pcap with a range of simulated potential attacks was used to replicate all current potential real attack scenarios.

6.2 Future Work

Several improvements can be implemented as part of future work.

Actual SDN environment with real traffic activities analysis

The research conducted in this study utilized a virtual environment and a simulated SDN environment for simulation and testing purposes. The dataset used for the research consisted of pre-recorded Excel files and PCAP files of network traffic, which provided real empirical data. However, one limitation of the study was that the available dataset was not up to date, and thus, it did not fully reflect current traffic behaviour, such as the increasing usage of social media, online streaming, and IoT connectivity, which were not adequately covered.

To address this limitation and further improve the implementation of the approach, it would indeed be beneficial to conduct testing in an actual SDN environment with real-time traffic activities. By using a live SDN setup, researchers can capture and analyse the latest network traffic patterns, including emerging trends and evolving attack vectors. This would provide a more accurate representation of the current network landscape and help evaluate the effectiveness of the proposed approach in real-world scenarios.

By incorporating real traffic activities, such as the growing prevalence of social media, streaming, and IoT devices, researchers can gather more comprehensive and up-to-date insights into the performance and behaviour of the proposed detection and mitigation methods. This would enable them to fine-tune and refine the approach to better address the challenges and demands of contemporary network environments.

In conclusion, while the research utilized pre-recorded datasets in a simulated SDN environment, conducting experiments in an actual SDN environment with real-time traffic would provide a more robust and accurate assessment of the approach. Incorporating the latest traffic behaviour and attack scenarios would contribute to the ongoing improvement and practical implementation of the proposed methodology.

Hybrid ML adoption analysis

The focus of the thesis was on the operation of a combination of a deep autoencoder and Random Forest algorithm for intrusion detection. During a work-in-progress presentation at an IEEE conference, valuable feedback was received, which pointed towards the direction of incorporating other machine learning classifications within the framework.

This feedback has provided a clear way forward for future work. The proposed framework can be expanded to include additional machine learning classifiers for intrusion detection. As new classifications and detection methods emerge and develop, they can be incorporated and explored within the framework. This continuous improvement and update of the framework would enable it to adapt to evolving cyber threats and enhance its effectiveness in detecting and mitigating intrusions.

By incorporating new classifications and detection methods, the proposed framework can benefit from the advancements in machine learning and intrusion detection research. This approach ensures that the framework

remains up-to-date and capable of handling emerging intrusion techniques and attack vectors.

In summary, the feedback received during the work-in-progress presentation at the IEEE conference has guided the future work of the thesis. The incorporation of new machine learning classifications and methods of detection within the framework is an area that can be explored and expanded upon. This approach allows for ongoing improvement and ensures that the proposed framework remains relevant and effective in the field of intrusion detection.

Bibliography

[1] N. McKeown *et al.*, “OpenFlow: Enabling Innovation in Campus Networks,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, p. 69, 2008.

[2] F. Ieee *et al.*, “Software-Defined Networking : A Comprehensive Survey,” vol. 103, no. 1, 2015.

[3] D. Kreutz, F. M. V. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” *Proc. Second ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw. - HotSDN ’13*, p. 55, 2013.

[4] T. Hurley, J. E. Perdomo, and A. Perez-Pons, “HMM-based intrusion detection system for software defined networking,” *Proc. - 2016 15th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2016*, pp. 617–621, 2017.

[5] S. Nanda, F. Zafari, C. Decusatis, E. Wedaa, and B. Yang, “Predicting Network Attack Patterns in SDN using Machine Learning Approach,” 2016.

[6] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep learning approach for Network Intrusion Detection in Software Defined Networking,” *Proc. - 2016 Int. Conf. Wirel. Networks Mob. Commun. WINCOM 2016 Green Commun. Netw.*, pp. 258–263, 2016.

[7] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, “Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks,” *2018 4th IEEE Conf. Netw. Softwarization Work. NetSoft 2018*, no. NetSoft, pp. 462–469, 2018.

[8] P. Wang, K. M. Chao, H. C. Lin, W. H. Lin, and C. C. Lo, “An Efficient Flow Control Approach for SDN-Based Network Threat Detection and Migration Using Support Vector Machine,” *Proc. - 13th IEEE Int. Conf. E-bus. Eng. ICEBE 2016 - Incl. 12th Work. Serv. Appl. Integr. Collab. SOAIC 2016*, pp. 56–63, 2017.

[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A Deep Learning Approach to Network Intrusion Detection,” *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 1, pp. 41–50, 2018.

[10] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” *Proc. - Conf. Local Comput. Networks, LCN*, pp. 408–415, 2010.

[11] C. Li *et al.*, “Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN,” *Int. J. Commun. Syst.*, vol. 31, no. 5, pp. 1–15, 2018.

[12] L. Barki, A. Shidling, N. Meti, D. G. Narayan, and M. M. Mulla, “Detection of distributed denial of service attacks in software defined networks,” *2016 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2016*, pp. 2576–2581, 2016.

[13] S. S. Mohammed *et al.*, “A New Machine Learning-based

Collaborative DDoS Mitigation Mechanism in Software-Defined Network," *Int. Conf. Wirel. Mob. Comput. Netw. Commun.*, vol. 2018–Octob, pp. 1–8, 2018.

[14] B. Krebs, "Who Makes the IoT Things Under Attack?," *Krebs on Security*, 2016. [Online]. Available: https://krebsonsecurity.com/2016/10/who-makes-the-iot-things-under-attack/. [Accessed: 04-Jul-2019].

[15] S. Kottler, "February 28th DDoS Incident Report," 2018. [Online]. Available: https://github.blog/2018-03-01-ddos-incident-report/. [Accessed: 04-Jul-2019].

[16] Verisign, "Q2 2018 DDOS TRENDS REPORT: 52 PERCENT OF ATTACKS EMPLOYED MULTIPLE ATTACK TYPES," 2018. [Online]. Available: https://blog.verisign.com/security/ddos-protection/q2-2018-ddos-trends-report-52-percent-of-attacks-employed-multiple-attack-types/. [Accessed: 04-Jul-2019].

[17] B. Deokar and A. Hazarnis, "Intrusion Detection System using Log Files and Reinforcement Learning," *Int. J. Comput. Appl.*, vol. 45, no. 19, pp. 28–35, 2012.

[18] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," p. 151, 2013.

[19] S. Matsumoto, S. Hitz, and A. Perrig, "Fleet : Defending SDNs from Malicious Administrators," *Proc. Second ACM SIGCOMM Work. Hot Top. Softw. Defin. Netw. - HotSDN '14*, pp. 103–108, 2014.

[20] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for openflow applications," p. 171, 2013.

[21] S. Wang *et al.*, "SECOD: SDN sEcure control and data plane algorithm for detecting and defending against DoS attacks," *IEEE/IFIP Netw. Oper. Manag. Symp. Cogn. Manag. a Cyber World, NOMS 2018*, pp. 1–5, 2018.

[22] J. Rene Beulah and D. Shalini Punithavathani, "A hybrid feature selection method for improved detection of wired/wireless network intrusions," *Wirel. Pers. Commun.*, vol. 98, no. 2, pp. 1853–1869, 2018.

[23] F. Zhang and D. Wang, "An effective feature selection approach for network intrusion detection," *Proc. - 2013 IEEE 8th Int. Conf. Networking, Archit. Storage, NAS 2013*, pp. 307–311, 2013.

[24] J. R. Beulah and D. S. Punithavathani, "Simple Hybrid Feature Selection (SHFS) for Enhancing Network Intrusion Detection with NSL-KDD Dataset," *Int. J. Appl. Eng. Res.*, vol. 10, no. 19, pp. 40498–40505, 2015.

[25] H. M. Imran, A. Bin Abdullah, M. Hussain, and S. Palaniappan, "Intrusions Detection based on Optimum Features Subset and Efficient Dataset Selection," *Int. J. Eng. Innov. Technol.*, vol. 2, no. 6, pp. 265–270, 2012.

[26] S. Revathi and A. Malathi, "Network Intrusion Detection Using Hybrid

Simplified Swarm Optimization and Random Forest Algorithm on Nsl-Kdd Dataset," *Int. J. Eng. Comput. Sci.*, vol. 3, no. 2, pp. 3873–3876, 2014.

[27] D. a. M. S. Revathi, "A Detailed Analysis on NSL-KDD Dataset Using Various Machine Learning Techniques for Intrusion Detection," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 1848–1853, 2013.

[28] H. Chae, B. Jo, S. Choi, and T. Park, "Feature Selection for Intrusion Detection using NSL-KDD," *Recent Adv. Comput. Sci. 20132*, pp. 184–187, 2013.

[29] H. F. Eid, A. Darwish, A. Ella Hassanien, and A. Abraham, "Principle components analysis and support vector machine based Intrusion Detection System," *Proc. 2010 10th Int. Conf. Intell. Syst. Des. Appl. ISDA'10*, pp. 363–367, 2010.

[30] A. S. Bhandari, "Feature Selection and Classification of Intrusion Detection System Using Rough Set," no. 2, pp. 20–23, 2013.

[31] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," *Arab. J. Sci. Eng.*, vol. 42, no. 2, pp. 425–441, 2017.

[32] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 2, pp. 289–302, 2016.

[33] N. Meti, D. G. Narayan, and V. P. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, vol. 2017–Janua, pp. 1366–1371, 2017.

[34] Q. Niyaz, W. Sun, and A. Y. Javaid, "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)," *ICST Trans. Secur. Saf.*, vol. 4, no. 12, p. 153515, 2017.

[35] Y. Chang, W. Li, and Z. Yang, "Network intrusion detection based on random forest and support vector machine," *Proc. - 2017 IEEE Int. Conf. Comput. Sci. Eng. IEEE/IFIP Int. Conf. Embed. Ubiquitous Comput. CSE EUC 2017*, vol. 1, pp. 635–638, 2017.

[36] Y. Y. Aung and M. M. Min, "An analysis of random forest algorithm based network intrusion detection system," *Proc. - 18th IEEE/ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2017*, pp. 127–132, 2017.

[37] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du, and M. Guizani, "SDN Controllers: Benchmarking & Performance Evaluation," pp. 1–14, 2019.

[38] U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar, and K. El-Khatib, "Evaluation of Deep Learning in Detecting Unknown Network Attacks," pp. 1–6, 2020.

[39] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based network anomaly detection," *Wirel. Telecommun. Symp.*, vol. 2018–

April, pp. 1–5, 2018.

[40] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” *ICISSP 2018 - Proc. 4th Int. Conf. Inf. Syst. Secur. Priv.*, vol. 2018–Janua, no. Cic, pp. 108–116, 2018.

[41] S. Ustebay, Z. Turgut, and M. A. Aydin, “Intrusion Detection System with Recursive Feature Elimination by Using Random Forest and Deep Learning Classifier,” *Int. Congr. Big Data, Deep Learn. Fight. Cyber Terror. IBIGDELFT 2018 - Proc.*, pp. 71–76, 2019.

[42] K. Kirutika, “Controller Monitoring System In Software Defined Networks Using Random Forest Algorithm,” 2019.

[43] R. Abdulhammed, M. Faezipour, H. Musafer, and A. Abuzneid, “Efficient network intrusion detection using PCA-based dimensionality reduction of features,” *2019 Int. Symp. Networks, Comput. Commun. ISNCC 2019*, 2019.

[44] S. Choudhury and A. Bhowal, “Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection,” *2015 Int. Conf. Smart Technol. Manag. Comput. Commun. Control. Energy Mater. ICSTM 2015 - Proc.*, no. May, pp. 89–95, 2015.

[45] M. Anbar, R. Abdullah, I. H. Hasbullah, Y. W. Chong, and O. E. Elejla, “Comparative performance analysis of classification algorithms for intrusion detection system,” *2016 14th Annu. Conf. Privacy, Secur. Trust. PST 2016*, pp. 282–288, 2016.

[46] “Mininet An Instant Virtual Network on your Laptop (or other PC),” 2018. [Online]. Available: http://mininet.org/. [Accessed: 13-Mar-2020].

[47] “Ryu SDN Framework,” 2017. [Online]. Available: https://osrg.github.io/ryu/. [Accessed: 13-Mar-2020].

[48] “Tensorflow,” 2017. [Online]. Available: https://www.tensorflow.org. [Accessed: 11-Oct-2018].

[49] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, “Network Intrusion Detection: Based on Deep Hierarchical Network and Original Flow Data,” *IEEE Access*, vol. 7, pp. 37004–37016, 2019.

[50] A. Binbusayyis and T. Vaiyapuri, “Identifying and Benchmarking Key Features for Cyber Intrusion Detection: An Ensemble Approach,” *IEEE Access*, vol. 7, pp. 106495–106513, 2019.

[51] T. Cisco and A. Internet, “Cisco: 2020 CISO Benchmark Report,” *Comput. Fraud Secur.*, vol. 2020, no. 3, pp. 4–4, 2020.

[52] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, “A security enforcement kernel for OpenFlow networks,” *HotSDN’12 - Proc. 1st ACM Int. Work. Hot Top. Softw. Defin. Networks*, pp. 121–126, 2012.

[53] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, “A SDN-oriented DDoS blocking scheme for botnet-based attacks,” *Int. Conf. Ubiquitous Futur. Networks, ICUFN*, pp. 63–68, 2014.

[54] S. M. Mousavi and M. St-Hilaire, “Early Detection of DDoS Attacks Against Software Defined Network Controllers,” *J. Netw. Syst. Manag.*, vol. 26, no. 3, pp. 573–591, 2018.

[55] M. M. Isa and L. Mhamdi, “Native SDN Intrusion Detection using Machine Learning,” *2020 8th Int. Conf. Commun. Networking, ComNet2020 - Proc.*, pp. 1–7, 2020.

[56] D. Sattar, A. Matrawy, and O. Adeojo, “Adaptive Bubble Burst (ABB): Mitigating DDoS attacks in Software-Defined Networks,” *2016 17th Int. Telecommun. Netw. Strateg. Plan. Symp. Networks 2016 - Conf. Proc.*, pp. 50–55, 2016.

[57] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, “SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN,” *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1545–1559, 2018.

[58] X. Huang, K. Xue, Y. Xing, D. Hu, R. Li, and Q. Sun, “FSDM: Fast recovery saturation attack detection and mitigation framework in SDN,” *Proc. - 2020 IEEE 17th Int. Conf. Mob. Ad Hoc Smart Syst. MASS 2020*, pp. 329–337, 2020.