A NEW PROPOSED STACKING GENERALIZATION MODEL FOR DETECTING DDOS ATTACKS IN SDN ENVIRONMENT

2023
MASTER THESIS
COMPUTER ENGINEERING

Tasnim Kamal ALASALI

Thesis Advisor
Assist. Prof. Dr. Omar DAKKAK

A NEW PROPOSED STACKING GENERALIZATION MODEL FOR DETECTING DDOS ATTACKS IN SDN ENVIRONMENT

Tasnim Kamal ALASALI

Thesis Advisor
Assist. Prof. Dr. Omar DAKKAK

T.C.
Karabuk University
Institute of Graduate Programs
Department of Computer Engineering
Prepared as
Master Thesis

KARABUK
June 2023

I certify that in my opinion the thesis submitted by Tasnim Kamal ALASALI titled "A NEW PROPOSED STACKING GENERALIZATION MODEL FOR DETECTING DDOS ATTACKS IN SDN ENVIRONMENT " is fully adequate in scope and in quality as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Omar DAKKAK ....
Thesis Advisor, Department of Computer Engineering

This thesis is accepted by the examining committee with a unanimous vote in the Department of Computer Engineering as a Master of Science thesis. June 9, 2023

<u>Examining Committee Members (Institutions)</u>		<u>Signature</u>
Chairman	: Assoc. Prof. Dr. Adib HABBAL (KBU)	....
Member	: Assist. Prof. Dr. Omar DAKKAK (KBU)	....
Member	: Assist. Prof. Dr. Yusuf Yargı BAYDİLLİ (HU)	....

The degree of Master of Science by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabuk University.

Prof. Dr. Müslüm KUZU ....
Director of the Institute of Graduate Programs

“I declare that all the information within this thesis has been gathered and presented in accordance with academic regulations and ethical principles and I have according to the requirements of these regulations and principles cited all those which do not originate in this work as well.”

Tasnim Kamal ALASALI

ABSTRACT

M. Sc. Thesis

A NEW PROPOSED STACKING GENERALIZATION MODEL FOR DETECTING DDOS ATTACKS IN SDN ENVIRONMENT

Tasnim Kamal ALASALI

Karabük University
Institute of Graduate Programs
The Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Omar DAKKAK
June 2023, 97 pages

The emergence of Software-Defined Networking (SDN) has revolutionized network infrastructure by providing greater control and operation over the network. The SDN controller, which serves as the operating system for SDN-based networks, executes various network applications and maintains network services and functionalities. However, the central control that SDN offers makes it vulnerable to Distributed Denial of Service (DDoS) attacks, which are the most common and critical attacks targeting both traditional and new-generation networks, including the Internet of Things (IoT), cloud computing, and fifth generation (5G) communication networks. Despite the plethora of traditional detection solutions available, DDoS attacks continue to increase in frequency, volume, and severity. To address this, machine learning is now widely used for rapid attack detection. This research proposes a predictive model for DDoS prediction in an SDN environment. The model is based on stacking various classifiers in two levels, namely the Base level and the Meta level, which combine diverse heterogeneous learners to produce robust model outcomes. Multiple metrics were used to evaluate the model's performance, including accuracy, precision, recall, F1-scores, and Area Under the ROC Curve (AUC) values. The predictive model achieved a 99% accuracy rate in prediction, with a precision score, sensitivity, and specificity all at 99%.

Key Words : Software-Defined Network (SDN), Security, DDoS Attacks, Machine Learning (ML), Stacking Classifier, RYU.

Science Code : 92430

ÖZET

Yüksek Lisans Tezi

SDN ORTAMINDA DDOS SALDIRILARINI TESPİT ETMEK İÇİN YENİ BİR İSTİFLEME GENELLEŞTİRME MODELİ

Tasnim Kamal ALASALI

Karabük Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Yrd. Doç. Dr. Omar DAKKAK
Haziran 2023, 97 sayfa

Yazılım Tanımlı Ağ Oluşturmanın (SDN) ortaya çıkışı, ağ üzerinde daha fazla kontrol ve operasyon sağlayarak ağ altyapısında devrim yarattı. SDN tabanlı ağlar için işletim sistemi olarak hizmet veren SDN denetleyicisi, çeşitli ağ uygulamalarını yürütür ve ağ hizmetlerini ve işlevlerini sürdürür. Bununla birlikte, SDN'nin sunduğu merkezi kontrol, onu Nesnelerin İnterneti (IoT), bulut bilgi işlem dahil olmak üzere hem geleneksel hem de yeni nesil ağları hedefleyen en yaygın ve kritik saldırılar olan Dağıtılmış Hizmet Reddi (DDoS) saldırılarına karşı savunmasız hale getirir ve beşinci nesil (5G) iletişim ağları. Mevcut geleneksel algılama çözümlerinin bolluğuna rağmen, DDoS saldırılarının sıklığı, hacmi ve şiddeti artmaya devam ediyor. Bunu ele almak için, makine öğrenimi artık hızlı saldırı tespiti için yaygın olarak kullanılmaktadır. Bu araştırma, bir SDN ortamında DDoS tahmini için tahmine dayalı bir model önermektedir. Model, sağlam model sonuçları üretmek için çeşitli heterojen öğrenicileri birleştiren Temel düzey ve Meta düzey olmak üzere iki düzeyde çeşitli sınıflandırıcıların istiflenmesine dayanmaktadır. Modelin performansını değerlendirmek için doğruluk, kesinlik, geri çağırma, F1 puanları ve ROC Eğrisi Altındaki Alan (AUC) değerleri dahil olmak üzere birden fazla metrik kullanıldı. Tahmine dayalı model, tamamı %99'da kesinlik puanı, duyarlılık ve özgüllük ile tahminde %99'luk bir doğruluk oranı elde etti.

Anahtar Kelimeler : Yazılım Tanımlı Ağ (SDN), Güvenlik, DDoS Saldırıları, Makine Öğrenimi (ML), İstifleme Sınıflandırıcı, RYU.

Bilim Kodu : 92430

ACKNOWLEDGMENT

I want to express my sincere gratitude to my thesis advisor, Dr. Omar DAKKAK, for his unwavering support, guidance, and encouragement throughout my research. Dr. Omar's expertise in my field has been invaluable, and I am deeply grateful for the time and effort he has invested in my work.

I would also like to thank my thesis committee members for their insightful feedback and helpful suggestions. Their expertise and constructive criticism have been instrumental in shaping my research.

I want to thank my family, particularly my parents, for their unwavering love and support. Their encouragement and belief in me have been a constant source of strength and inspiration.

Finally, I thank my husband, Dr. Akram ALHAMAD, for his unwavering support, encouragement, and love throughout my graduate studies. His patience, understanding, and belief in me have been a constant source of strength and inspiration. Without his support, I would not have been able to complete this thesis.

I would also like to thank my baby, Sanm, for bringing joy, laughter, and inspiration to my life. Her smiles, hugs, and giggles have been a bright spot in my days, and her presence has motivated me to keep going even when the going got tough. To my husband and baby, thank you for being my rock and being there for me through every step of this journey. Your unwavering love and support mean the world to me, and I am grateful for every moment we have spent together.

CONTENTS

Page

Page

LIST OF FIGURES

LIST OF TABLES

Page

ABBREVIATIONS INDEX

ABBREVIATIONS

API	: Application Programming Interface.
AUC	: Area Under the ROC Curve
AWS	: Amazon Web Services
CLI	: Command Line Interface
CM	: Confusion Matrix
CNN	: Convolutional Neural Network
DDoS	: Distributed Denial of Service Attack
FN	: False Negatives
FP	: False Positives
FPR	: False Positive Rate
GRU	: Gated Recurrent Unit
HITL	: Human-In-The-Loop
ICMP	: Internet Control Message Protocol
IDS	: Intrusion Detection Systems
IOT	: Internet of Things
KNN	: K-Nearest Neighbors
LR	: Logistic Regression
LSTM	: Long Short-Term Memory
ONOS	: Open Network Operating System
ML	: Machine Learning
MLP	: Multilayer Perception
NB	: Naïve Bayes
NBI	: Northbound Interface
NFV	: Network Function Virtualization
ONF	: Open Networking Foundation
OVS	: Open Virtual Switch

RF	: Random Forest
RFC	: Request for Comments
RFE	: Recursive Feature Elimination
ROC	: Receiver Operating Characteristic
RPS	: Requests Per Second
SBI	: Southbound Interface
SDN	: Software-Defined Networking
SOM	: Self-Organizing Maps
SVM	: Support Vector Machine
TCP	: Transmission Control Protocol
TLS	: Transport Layer Security
TN	: True Negatives
TP	: True Positives
TPR	: True Positive Rate
UDP	: User Datagram Protocol
VLAN	: Virtual Local Area Networks

PART 1

INTRODUCTION

The ubiquitous presence of computer networks has made them an integral part of our daily routines, ranging from government and commercial enterprises to individuals [1]. These networks, composed of numerous routers, switches, and middleboxes, run on complex protocols, making operating, and maintaining a secure and seamless infrastructure quite daunting. As the demand for ubiquitous connectivity from end users continues to rise, keeping up with network administration tasks has become increasingly tricky [2]. Integrating software into the daily routine, with the wide adoption of smart mobile devices and the recent emergence of the Internet of Things (IoT), has further heightened the race to keep up with network administration tasks. Network administrators are thus responsible for configuring vendor-specific devices and setting policies for effective and reliable operation to accommodate the ever-changing network environment. Unfortunately, managing networks and dynamic responses to events and applications is arduous and error prone. The service providers, however, must meet consumers' growing traffic and data rate demand by investing in more extensive and faster links and edge routers, despite the slow revenue growth [3] [4]. Network administrators and vendors face significant challenges, making it essential to optimize and innovate current network design and architecture with cost, programmability, and network robustness in mind, to meet the growing demands of users.

The paradigm of Software-Defined Networking (SDN) has become increasingly popular in recent times. SDN-based networks replace traditional data centers and operator networks because they provide a more reliable, flexible, and secure environment [5], [6]. As a result, deploying SDN in cloud computing environments and data centers provides a flexible and reliable network architecture. The primary innovation behind SDN is the separation of control and data planes. SDN provides intelligent centralization through controllers that manage the forward packet devices. The well-designed configuration of these devices, such as Open-Flow, is also essential [7], [8]. In an SDN, switches only forward logic, while decision-making and control logic is carried out by software in an SDN controller. The SDN controller is responsible for creating new network flow policies and instructing switches to follow the guidelines maintained in a flow table [9].

Moreover, SDN architecture represents a groundbreaking and innovative approach to network management [10]. SDN upends conventional network paradigms by assigning switches a fundamentally different role. Instead of traditional packet processing, switches perform a matching operation against their forwarding tables, delegating unmatchable packets to the controller for processing. The controller serves as the veritable operating system of the SDN, responsible for handling packet processing and determining whether to forward or discard each packet. This process thus bifurcates the traditional forwarding and processing planes of network architecture, yielding an agile and highly programmable network environment. The SDN architecture lends itself to forming a network comprising many controllers, each of which interfaces with a network of switches, yielding a complex mesh of individual network slices. This granular segmentation offers a unique opportunity to safeguard each slice against the perils of DDoS attacks. Notably, if the connection between a given controller and its associated switches is severed, the processing plane of the entire network slice will falter. The attendant consequence of such an occurrence is the virtual extinction of packet processing capability within the affected slice, leading to a pronounced collapse of the overarching SDN architecture.

The controller's accessibility can be jeopardized by various factors, including but not limited to the threat of a DDoS attack [11]. This attack operates by inundating a host or group of hosts within a given network with packets. These packets are often furnished with falsified source addresses, eluding conventional switch-matching protocols and necessitating the controller's intervention. However, the cumulative load of legitimate and DDoS-sourced packets can exert a crippling strain on the controller's resources, depleting them in an unrelenting torrent of processing. This protracted overuse renders the controller inaccessible to legitimate incoming packets and potentially causes the entire SDN architecture to collapse. Even the existence of a backup controller is not a panacea, as it too would be confronted with the same problem in the event of a DDoS attack [12], [13]

The present chapter illustrates the problem statement by scrutinizing the menace posed by DDoS Attacks in the SDN architecture. After that, the motivation driving the research and its objectives will be expounded. Subsequently, the research contributions will be delimited, followed by a delineation of the organizational structure of this thesis.

1.1. MOTIVATION

As such, the emergence of SDN has brought great potential to overcome the need for flexible, secure, reliable, and well-managed next-generation networks [14]. SDN transforms switches into simple data forwarding devices by separating networks' control and data plane, while a logically centralized controller controls network management. This remarkable feature of SDN provides a programmable, vendor-agnostic, cost-effective, and innovative network architecture.

The future of SDN depends on its acceptance and deployment. Although technology deployment typically takes years before becoming available to end-users due to standardization processes and Requests for Comments (RFC), it remains unclear whether the same will hold for SDN. Thus far, researchers have conducted experiments on campus networks, leading to the deployment of SDN on a small scale and, ultimately, the proposal of a new network architecture, ETHANE, for enterprise networks[15][16]. Figure 1.1 illustrates the conventional development trend of technology [15].

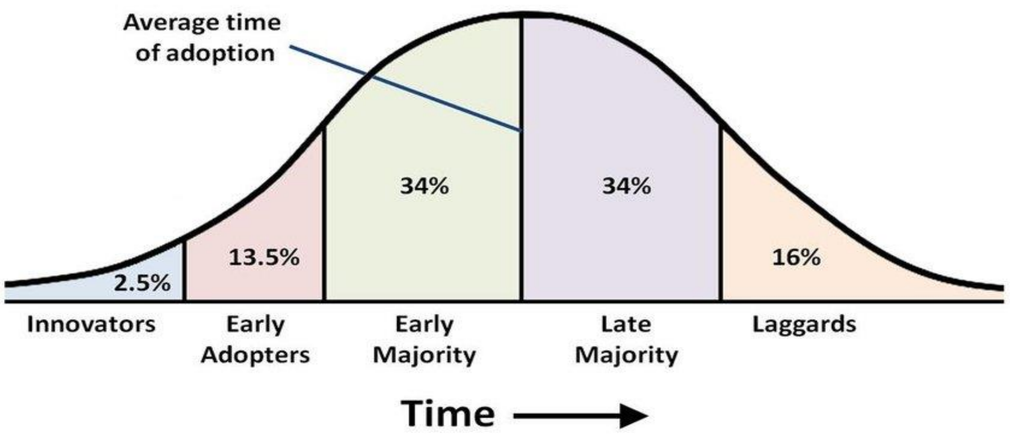


Figure 1.1. Diffusion of innovation [17].

The trajectory of SDN development may deviate slightly from the traditional path as manufacturers and vendors of network equipment are increasingly drawn to the potential and benefits of this technology, as demonstrated by Google's deployment of B4 in their WAN data center [18], VMware's NSX virtualization platform [19], and the Linux Foundation's collaborative project, OpenDaylight [20]. However, despite its critical features of programmability, flexibility, universal connectivity, and centralized control, security issues remain a significant concern for SDN. These security vulnerabilities include an increased potential for DDoS attacks due to network device controller centralization and flow table limitation [21]. Additionally, the abstraction of flows and underlying hardware resources renders it easier to gather intelligence that can be used for further exploitation and malicious network reprogramming by a bad actor [22], [3].

Security is a paramount concern in computer networks, as illicit access to these systems would entail virtually unfettered control over the company's resources. Virtual Local Area Networks (VLANs) were conceptualized as physically segregating distinct organizational functions to preclude direct interactivity and bolster legacy network security. In the context of SDN, network virtualization is enabled by implementing rigorous criteria – such as packet header fields – which allow for the high-level abstraction necessary to facilitate the efficient sharing of physical network infrastructure among multiple users concurrently. Network Function Virtualization (NFV), a closely related and complementary concept to SDN, further catalyzes the programmability of essential network functions, including DNS and caching, thereby fostering an environment ripe for service innovation and the introduction of novel network services [23].

In SDN, the controller communicates with the forwarding plane via the southbound API utilizing a secure transport layer service. Each switch has a designated flow table that contains specific flow information. When a packet arrives at the switch, and its header fields fail to conform to the established flow table entries, the packet is automatically routed to the controller as a "packet-in" message. The controller then processes the packet and returns a "packet-out" or "flow-mod" message containing the requisite flow rules installed in the flow table. Subsequent packets with similar characteristics will then be efficiently acted upon without any reference to the controller.

Despite its impressive innovations, several components of the SDN-based network environment pose additional security threats to the SDN controller. Thus, the security of the SDN controller is becoming a key challenge for developing future SDN-based networks [2]. The centralization of the controller has the potential to simplify network management, yet it poses a significant security threat [24]. Malicious actors can launch voluminous requests to the controller to overwhelm it and render it offline, causing service unavailability and disabling the SDN. Moreover, SDN is vulnerable to DDoS attacks, which can target the controller and flow tables in the switch, clogging them with data and rendering the entire network inoperable [24]. These vulnerabilities underscore the necessity of having backups for controllers. However, the backup controllers are equally susceptible to attacks, which can result in their compromise and eventual shutdown based on the amount of traffic directed at them [12], [13]. Therefore, there is a compelling need for a proactive and responsive system capable of detecting and mitigating such attacks in real-time.

1.2. PROBLEM STATEMENT

SDN separates the data and control planes, simplifying network management and enabling programmability[25]. However, the centralized nature of the controller in

SDN networks makes it vulnerable to DDoS attacks, which can bring down the entire network [24]. These attacks flood victims with traffic from multiple compromised systems using spoofed source IP addresses, making it difficult to block them based solely on IP addresses. Distinguishing legitimate user traffic from attack traffic becomes challenging when the sources are dispersed across the Internet. Effective DDoS detection and response methods are crucial for network operations, with SDN networks being particularly vulnerable [12] [2]. The controller's central role makes it an attractive target for attackers, and the attack packets overwhelm switches and the controller, depleting their resources and potentially causing network failure [13]. DDoS attacks are more damaging and faster in SDN networks compared to traditional networks, emphasizing the need for reliable detection methods and prompt action [21], [26]. In large data centers where SDN networks are prevalent, it is essential to identify targeted network segments quickly to facilitate mitigation measures. While controllers are designed with backups and ample memory, switches have limited resources, making them more susceptible to attacks. Hence, implementing lightweight provisional methods to prevent switch breakdown at the first signs of an attack is crucial. The detection method should impose minimal additional load on the controller. Consequently, this study proposes a novel approach designed explicitly for detecting DDoS attacks in SDN environments, providing distinct benefits over existing solutions.

1.3. RESEARCH QUESTIONS

This thesis aims to answer the following questions:

1. What types of DDoS attacks target SDN, and how can they be detected?
2. How can a realistic SDN dataset with regular and attack traffic be generated?
3. How can Stacking Classifiers effectively detect DDoS attacks in SDN?

1.4. RESEARCH OBJECTIVES

In the modern interconnected world, security breaches propagate rapidly and can affect many networks across the globe within a matter of hours or days. Even with secure

perimeters, organizations often face internal security breaches. Security-focused SDN architecture has been developed to prevent malicious users from exploiting vulnerabilities in the network. Therefore, this study aims to propose an intelligent machine learning based approach utilizing a stacking classifier model to detect DDoS attacks in SDN environments.
To achieve this aim, the study defines the following objectives:

1. Identifying DDoS attacks on SDN and reviewing detection methods.
2. Designing an SDN topology to generate a dataset with regular and attack traffic.
3. Developing a stacking classifier model consisting of multiple base classifiers for DDoS detection.

1.5. SCOPE OF THE RESEARCH

This research investigates the effectiveness of machine learning approaches in detecting DDoS attacks in SDN networks. It also focuses on identifying DDoS attacks in SDN networks and evaluating existing SDN security solutions related to these attacks. The Stacking Classifier technique will be introduced to detect DDoS attacks in SDN networks, and its effectiveness in improving the classification performance will be evaluated.

Real SDN traffic datasets were used to evaluate the proposed approach for detecting DDoS flooding and attack-free network traffic, as discussed in Chapter 4. It is important to note that all DDoS flooding attacks launched during the research were orchestrated from internal networks. As a result, DDoS attacks launched from external networks are not covered and are beyond the scope of this work.

1.6. ORGANIZATION OF THE THESIS

The present thesis comprises six chapters, each with a specific focus. Chapter 1 serves as an introduction, outlining the concept of Software-Defined Networking (SDN) and presenting the research objectives and questions, as well as the scope of the research.

Chapter 2 provides an overview of SDN and its components, followed by an examination of Distributed Denial of Service (DDoS) attacks and their various types. Furthermore, the chapter discusses the impact of DDoS attacks on controllers.

Chapter 3 reviews the literature on DDoS detection and mitigation within SDN.

Chapter 4 proposes an end-to-end intrusion detection approach for SDN-based networks.

Chapter 5 presents an experimental evaluation of the proposed intrusion detection system.

Finally, Chapter 6 summarizes the proposed method and its simulation results. The chapter also outlines the solution's limitations and suggests future work.

PART 2

THEORETICAL BACKGROUND

The Open Networking Foundation (ONF) [28] is a non-profit organization devoted to the standardization, research, and commercialization of software-defined networking (SDN). ONF has supplied the most explicit and widely accepted description of SDN: SDN is a new network architecture separating network control from forwarding and can be programmed directly [29] [30]. This isolation is done by rebuilding the network so that the switch forwarding instructions are received from the controller rather than spending its resources to process incoming packets. The Openflow protocol orchestrates the SDN architecture. Consequently, OpenFlow switches, a controller, and a secure connection between the controller and the switches make up this architecture.

This chapter will overview SDN, the OpenFlow protocol, and its specification. Lastly, it will cover Distributed Denial of Service (DDoS) Attacks in the SDN environment.

2.1. SOFTWARE DEFINED NETWORK (SDN)

First, it is necessary to mention the traditional networks to note the change brought by the SDN networks. In traditional network systems, network devices require powerful hardware for processing and making decisions, saving routing tables and access lists, and robust hardware for passing packets, as shown in Figure 2.1.

On the other hand, the network devices each include their own unique routing tables and access lists. This leads to the decentralization of the network, which in turn implies that each device must be set independently. In addition, it becomes more difficult to identify defects and issues in big networks as the number of network devices continues to grow. In addition, when a new device is introduced to the network, all of the existing devices on the network need to have their settings adjusted so that they can function in tandem with the newly added device. The implementation layer, which is the physical forces of the network, is separated from the decision-making layer, which is the brains of the network, by software-defined networking (SDN), which adds a control center to the network (Controller) that provides a comprehensive view of the network devices and the rules that are imposed on them.

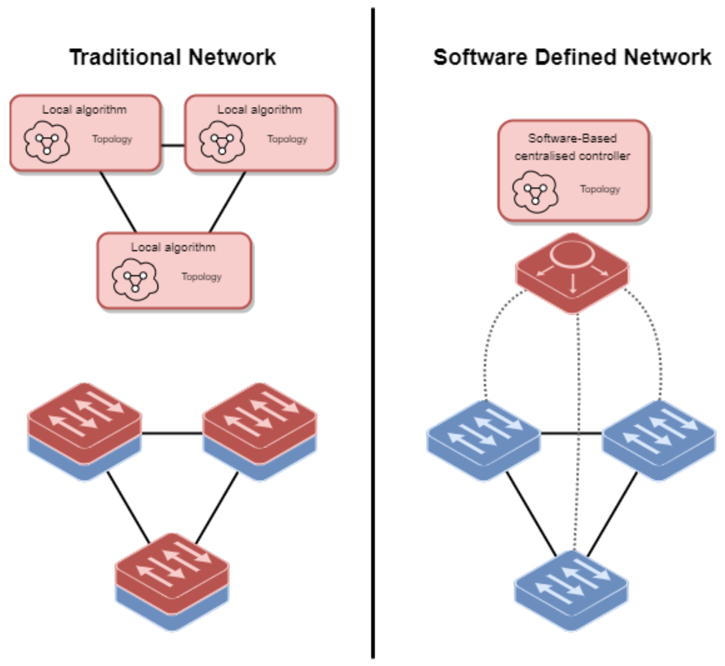


Figure 2.1. SDN vs. traditional networks.

However, SDN is a new network architecture significantly improves network performance [2]. It provides the flexibility to control the entire network without the need to access all the devices, which are usually geographically far apart. Figure 2.2 shows the main features of these networks. Specific criteria have been unmet by traditional network architecture, such as virtualization, accessibility, dynamic management, high bandwidth, high connection speed, and cloud computing [31]. As a result, SDN, which has a flexible, programmable, and dynamic network design, has emerged as a viable alternative to traditional networks [28].

SDN is characterized as:

1. Programmable: Decoupling forwarding and control plans makes this possible [28].
2. Agile: the abstraction of the control layer from the forwarding layer allows network engineers to adjust traffic flow to fit their present demands.
3. Programmatically Configured: SDN provides network engineers with the ability to set up, administer, protect, and quickly optimize network resources, utilizing dynamic, custom-built SDN programs that are vendor-agnostic and built by themselves.
4. Central Management: SDN employs the advantage of the global view that the controller must facilitate network management. Also, it is possible to simultaneously reconfigure several devices[32].

SDN is vendor-agnostic and utilizes open-source standards, simplifying network design and operation because SDN controllers provide commands rather than various vendor-specific devices and protocols.

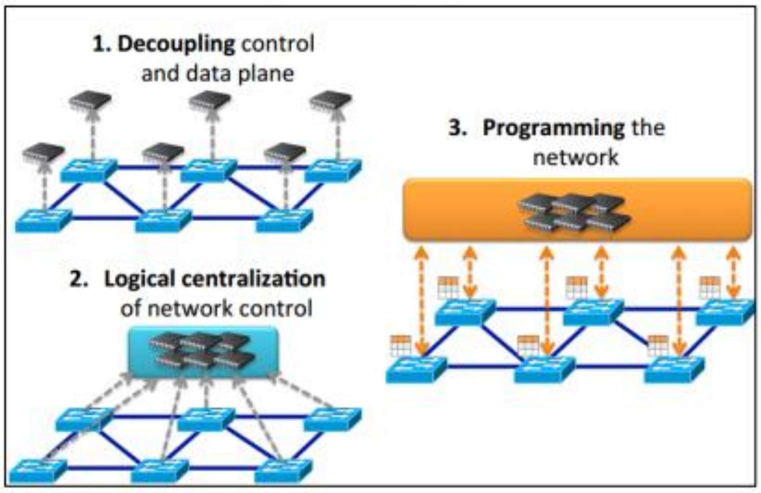

1. Decoupling control and data plane
3. Programming the network
2. Logical centralization of network control

Figure 2.2. SDN features.

Accordingly, as summarized in Table 2.1, it can be distinguished between the software-defined networks and the traditional networks in that the SDN can be configured during the network operation easier. In addition, the control's centralization feature helps reduce the cost of operating and managing the network. Also, when designing the network, it is necessary to focus on the services provided by the network

Accordingly, as described in Table 2.1, it is possible to differentiate between software-defined networks and traditional networks in that the SDN may be set during the operation of the network more easily. This is one of the key differentiating factors. In addition, the centralized function of the control helps cut down on the expenses associated with maintaining and administering the network. It is also necessary to focus on the services provided by the network control center when designing the network. This is true regardless of the types and manufacturers of switches that are used in the network; this is because SDN networks operate on the OpenFlow protocol, which has been agreed upon by the manufacturers. When it comes to conventional networks, configuring them while they are operational might be difficult, and doing so entails pausing the network while the initialization process is carried out. Additionally, in order to exercise control, conventional networks need the individual configuration of each device. This is the conundrum that conventional networks face when it comes to the detection of defects. On the other hand, detecting faults requires a lot of time.control center, regardless of the types and manufacturers of switches used in the network; This is because the SDN networks operate on the OpenFlow protocol agreed by the manufacturers. As for traditional networks, it isn't easy to configure when it is working, and this requires stopping the network at the time of initialization. Also, in terms of control, traditional networks must configure each device separately. Regarding detecting errors, this is the dilemma of traditional networks. In contrast, finding errors takes a long time.

Table 2.1. Comparisons of SDN and traditional networks.

Features	Software-Defined Networks	Traditional Networks
Programmability and Data and Control Plane Isolation	Yes	No, with complex network control requiring different techniques for each problem
Configuration	Automated with centralized authentication	Manual configuration is error-prone [33]
Performance	Global control that is dynamic, using input from other layers	Limited information and a rather endless configuration
Innovation	A straightforward software implementation for novel concepts, a comprehensive testing environment that allows for isolation, and an upgrade system that makes deployment simple	Hardware implementation for novel concepts is complex, there is a limited testing environment, and the standardization process is lengthy

2.2. SDN ARCHITECTURE

A typical representation of an SDN architecture consists of three layers. Northbound APIs connect applications to the control layer, whereas southbound APIs connect the control layer to the data plane, as shown in Figure 2.3.

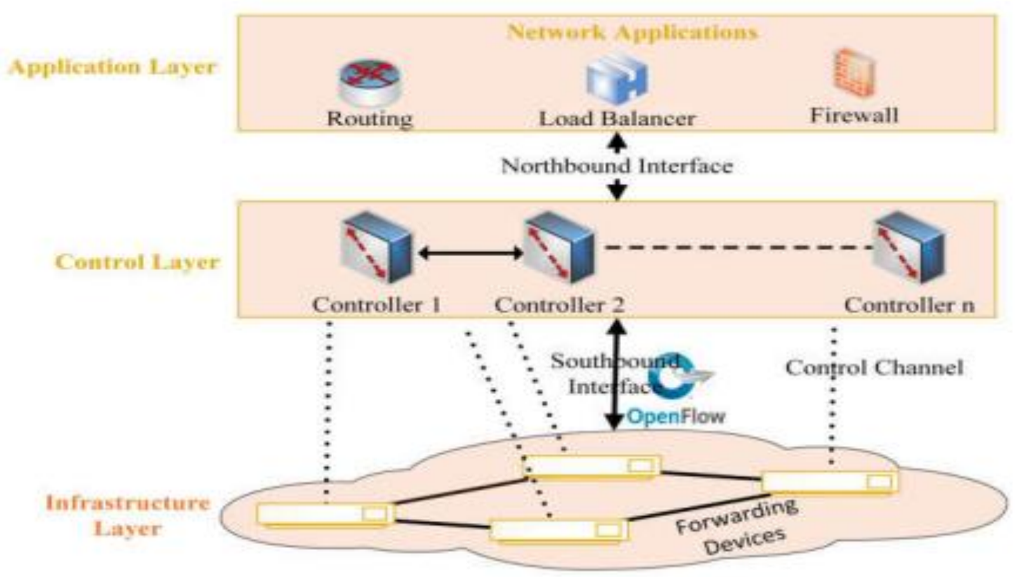


Figure 2.3. SDN structure (Wazirali et al., 2021).

1. Application Layer: this layer includes typical network applications and operations that businesses use, such as Intrusion detection systems (IDS), load balancing, firewalls, etc... Moreover, SDN networks replace a traditional network's specialized equipment with an application that can build a single view of the network by collecting information from the controller for decision-making purposes and regulating data plane behavior.
2. Control Layer: This plane has been referred to as the "brain" of the SDN [28]. The intelligence needed for this layer is provided by a centralized SDN controller. This controller is the software that takes instructions or commands from the application layer and then distributes those instructions or commands to the networking components. Additionally, it gathers statistical data by employing open APIs in order to give a perspective of the network and its topology that is as current as possible [23]. Despite the fact that the forwarding

engine is located in the switches (in the Data Plane), all of the network management tasks, such as monitoring traffic and routing it, are carried out by a centralized software-based controller [21]. This technology makes it possible to make dynamic changes to the routing rules that are implemented in network devices such as switches and routers. In addition, approaches for the detection and mitigation of network-based attacks are able to be built with such capabilities.

3. The Data Layer is the infrastructure layer, and it consists of both physical and virtual network devices in its Data Plane. Historically, both the control and data planes of traditional networks were included inside the same physical device. Nevertheless, only the data plane is present in SDN-based network devices. As a consequence of this, the most important responsibility of these network devices is to transmit data. Because of this, the method for forwarding is made to be very effective.

Meanwhile, SDN Controller uses APIs to control and communicate with these higher and lower layers.

1. Northbound Interface (NBI): SDN applications depend on the controller to inform them of the network infrastructure state so that they may determine what resources are accessible. Furthermore, the SDN controller may verify that application traffic is routed under network administrators' policies. However, the control layer orchestrates how these apps are allowed access to the network's resources. It also utilizes its intelligence to determine the best path for the application in terms of latency and security requirements. The orchestration process is fully automated and does not require any manual configuration. Also, it varies considerably from one controller to another. This is because each controller provides and uses its version of the NBI [34].

2. Southbound Interface (SBI): The SDN controller communicates with network equipment in the data layer through southbound APIs, such as routers and switches. However, there are several SBIs available for controllers to communicate with devices. The most used SBI protocol in SDN is OpenFlow [34]. To establish a

connection with the switches, controllers listen to TCP port number 6653 for OpenFlow V1.3+. The OpenFlow protocol uses different messages to regulate communication between switches and controllers. Flow Mod, Packet IN, and Packet OUT are the three most used messages by OpenFlow to handle flow rules.

2.3. OPEN FLOW

OpenFlow, considered one of the primary standards for the southbound interface in SDN, was initially designed by researchers at Stanford University in 2008 [35]. Currently, the Open Networking Foundation (ONF) manages this protocol [28] with version 1.5 being the most recent one used in the industry. OpenFlow provides more straightforward communication between controllers and physical devices on the data plane [12] built on top of the Transmission Control Protocol (TCP) and utilizing Transport Layer Security (TLS) to secure its communications [34]. The communication between switches and controllers uses OpenFlow messages, and comprehending these messages and events is essential for building SDN scripts to modify the forwarding layer's behavior.

2.3.1. Open Flow Events

The OpenFlow protocol defines a set of events that can be triggered during the communication between the controller and the switches in an SDN network. These events include the connection up event, which occurs when a control channel is established between the controller and the switch, and the connection down event, which occurs when the connection is terminated. The flow_removed event is triggered when the switch informs the controller that a flow entry has been deleted due to an idle or hard timeout. The Packet_in event is launched by the controller when a packet received by the switch does not match any of the flow entries, and it is critical for DDoS detection, particularly for packet spoofing and ACK DDoS attacks.

Additionally, statistical events are triggered when the controller receives an OpenFlow statistics response message from the switch, such as the ofp_stats_reply message. These events, including the ofp_aggregate_stats_reply, ofp_port_stats, ofp_table_stats, ofp_flow_stats, and ofp_queue_stats, allow the controller to retrieve various statistics from the switches, such as flow statistics, which are essential for DDoS detection. By analyzing the number of flows acquired via stats request from the controller, DDoS attacks can be predicted in the SDN network. Therefore, understanding OpenFlow events is crucial for building effective DDoS detection mechanisms in SDN networks.

2.3.2. Open Flow Messages

OpenFlow messages facilitate communication between controllers and switches and can trigger specific events. Some key OpenFlow messages are described below:

1. Features Request: A message sent by the controller to the switch, consisting solely of the OpenFlow header.
2. Features Reply: The switch's response to the controller's ofpt_features_request message.
3. Packet Out: A message the controller sends to the switch, instructing it to send, enqueue, or discard a packet. This message includes attributes such as buffer ID, in_port, actions, and data.
4. Flow Modification Message: A message sent by the controller to the switch that indicates how to modify the flow table. The hard and idle timeout fields are critical because they control how quickly flows expire, allowing the controller to delete questionable flows.
5. Port Modification Message: A message used to change the behavior of physical ports.
6. Statistics Messages: Messages related to flow statistics, including Aggregate Flow Statistics, Port Statistics, Individual Flow Statistics, and Table Statistics.
7. Packet-In Message: Sent by the switch to the controller when packets arrive that don't match any fields. The controller then decides what actions to take, such as forwarding the packet to a specific port, dropping it, or altering its headers.
8. Flow Removed Message: Sent by the switch to the controller to indicate that a flow has been deleted. This message includes fields such as the OpenFlow header, match, cookie field, packet count, byte count, and duration, which can

be measured in nanoseconds or seconds. The 'reason field' in this message specifies why the flow was removed, such as because of idle timeout or hard timeout.

SDN allows for deploying various applications, such as traffic engineering, network security, virtualization, routing, network monitoring, and load balancing. SDN's ability to enable the network to develop at the speed of software rather than hardware has made all these features possible.

2.3.3. Open Flow Switch

The Openflow switch comprises a flow table, or multiple tables, and a secure channel to the controller. Each table contains a match field, counters, and instructions for each entry. The matching process of the switch covers various packet header fields, which are specified in Table 2.1. One of the fields in the table is the metadata field, which is defined as a maskable register that carries information from one table to another in cases where multiple tables are used. This enables the transfer of header information between tables. It is common for switches to have multiple pipelined tables, with the packet moving from one table to another until a match is found.

When a packet enters the switch, it is compared against all existing flows in the tables. The corresponding action is executed if a match is found and the associated counter is updated. The counters cover various components in the switch, such as counters per flow entry, per table, per port, per queue, and other areas. For example, the duration countermeasures the amount of time a flow spends in the table. The controller can access some of these counters for different purposes. However, it is essential to note that not all counters are used, as controllers are developed or configured to match the vendor's needs. Additionally, version 1.3 of the specifications allows counters to be disabled.

The packet is sent to the controller if a match is not found. In version 1.3 of the specifications, if no field can be matched, the packet is dropped since its header does not contain any fields specified in Table 2.2. Therefore, the packet is considered invalid or illegal.

Table 2.2. Packet header match fields.

Header Field
Ingress Port
Metadata
Ether src
Ether dst
Ether type
VLAN id
VLAN priority
MPLS label
MPLS traffic class
IPv4 src
IPv4 dst
IPv4 proto / ARP opcode
IPv4 ToS bits
TCP / UDP/ SCTP src port ICMP Type
TCP / UDP / SCTP DST ICMP code

In the context of SDN, packets can be either sent entirely to the controller or the switch can buffer only the header information, which is the default mode of operation. The packet is encapsulated and marked as an OFPT_PACKET_IN message in the former case, called Packet-In. The controller is responsible for processing these packets, considering the number of switches, time of day, packet length, and priority. After processing, the controller sends a response with an action to be taken for that packet and subsequent packets from the same source. At this point, the switch's role is limited to forwarding the packets.

The controller can issue a set of actions to the switch, including forwarding, dropping, queuing, quality of service, and modifying specific fields such as VLAN tag, MAC address, or IP address. Forwarding and dropping are the primary actions while queuing and modifying fields are optional. The controller sets the desired action for the packet and returns it to the switch through a Packet-Out message.

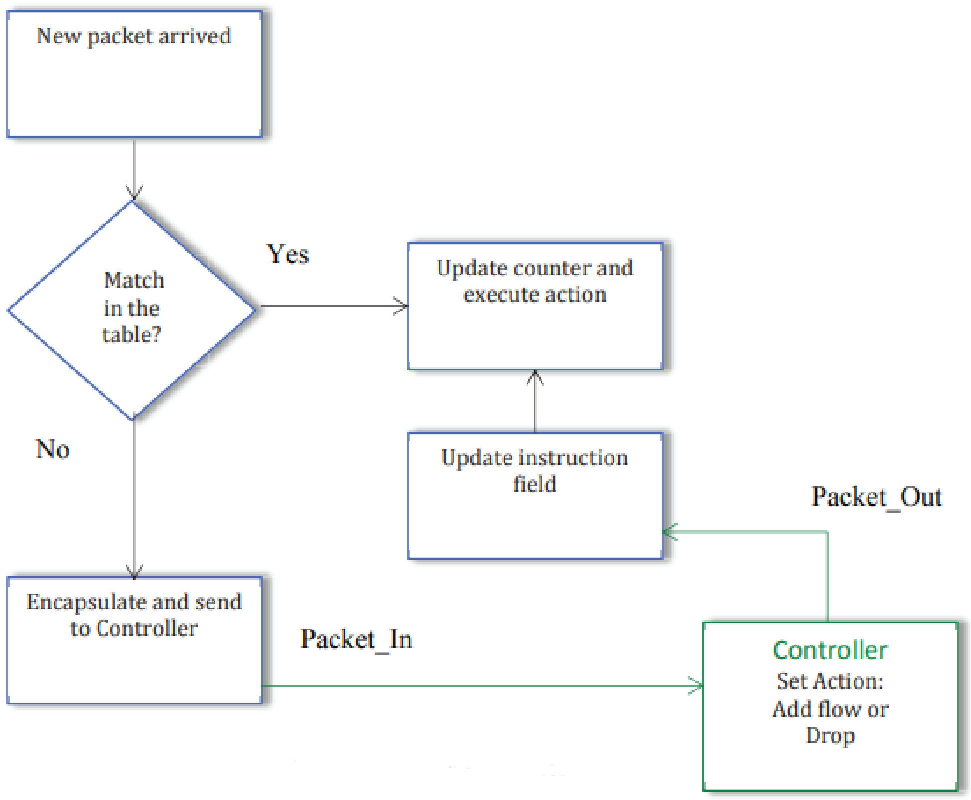

New packet arrived
Match in the table?
Yes
Update counter and execute action
Update instruction field
No
Packet_Out
Encapsulate and send to Controller
Packet_In
Controller
Set Action: Add flow or Drop

Figure 2.4. Flow entry process [12].

Figure 2.4 illustrates the flow entry process. If a packet is marked with a drop action, a corresponding flow entry will be created, and any packet matching that flow will be dropped. Without incoming packets, the flow entry will eventually expire and be removed from the table after a predetermined time.

2.4. DISTRIBUTED DENIAL OF SERVICE ATTACK

DoS attacks aim to prohibit legitimate users from accessing a service by overwhelming servers, networks, or service resources. Furthermore, it can not only restrict legal users from gaining access and utilizing network resources but also destroy the network [36][37]. DoS attacks can be coordinated, including many clients, to make the attack more effective. This attack is known as DDoS because it targets a service via distributed sources. Because these distributed resources are typically geographically separated, identifying the head of the attack and blocking malicious traffic can be difficult. While DDoS attacks necessitate client coordination, botnets are commonly used to carry them out. A botnet is a collection of zombie computers (bots) controlled by a central command and control server.

The DoS/DDoS attacks represent a persistent and severe threat to both traditional and modern network environments, including the Internet of Things (IoT) [38], cloud computing[39] [40], and fifth-generation (5G)[41], [42] communication networks. In recent years, these attacks' frequency, volume, and severity have continued to escalate. According to the Kaspersky DDoS attacks report for Q4 2020 [43]. Compared to 2019, the number of DDoS attacks in 2020 increased three times. Likewise, according to Amazon Web Services (AWS) research, the most significant DDoS attack to date was 2.3 Tbit/s in Q1 2020 [44]. Also, the attacks have become more sophisticated and intelligent over time, posing a more significant challenge for network security [45]. According to [46] a DDoS attack consists of four components, as illustrated in Figure 2.5:

1. The primary attacker oversees all aspects of the attack preparation and intelligence.
2. The handlers, or masters, are infected hosts that run special programs to control numerous agents.
3. The infected hosts run the attacker's malware and generate packet streams to target victims. These agents, also known as zombie hosts, are frequently unknown to the owner of the agent system.
4. The intended victim or destination.

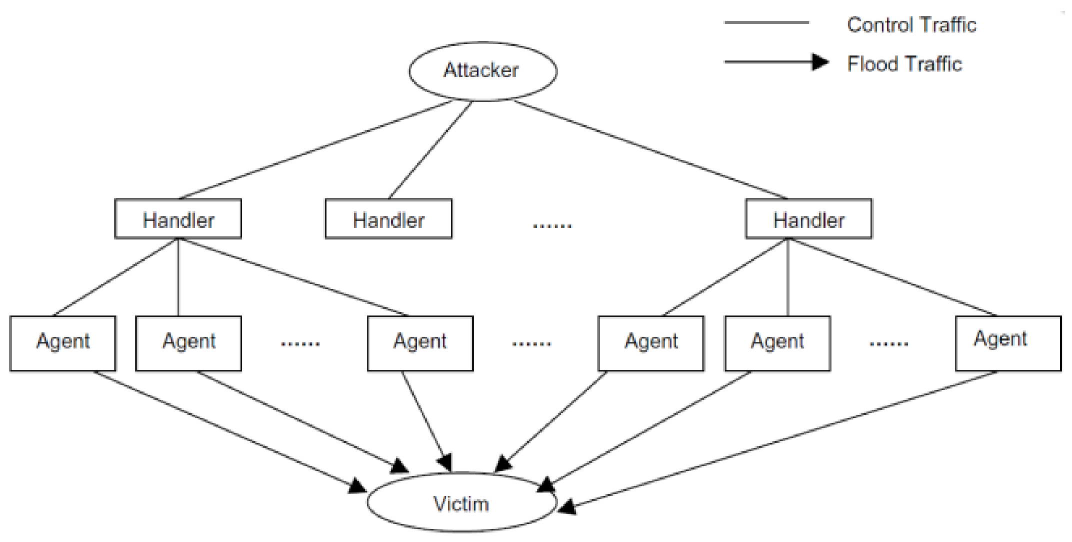


Figure 2.5. Text text.

The attacker identifies devices that have security vulnerabilities that can be exploited to gain access to them over the internet. To ensure the effectiveness of the attacking malware, the attacker must verify that the selected hosts meet the system requirements. This task is facilitated by the availability of readily accessible programs, thereby simplifying the attacker's job. The attacker can exploit the detected vulnerabilities and execute its code on the host. The codes are designed in a format that utilizes only a fraction of the host system's resources to evade detection by its owners. In addition, the codes are developed to be as undetectable as possible by the host system's programs. The attacker needs to communicate with the handlers to assess the eligibility of hosts for participation in the attack or to manipulate their codes. One or more handlers are responsible for managing each host, and communication between the attacker and the handlers is facilitated by UDP, TCP, or ICMP protocols. The attacker devises a plan for the attack, including the target victim's address and the parameters of the attack traffic, such as the port, duration, TTL, and traffic type. To avoid detection, all these characteristics may be altered during the attack stage.

2.4.1. DDOS Attack Types

It is vital to categorize these attacks in order to have a better knowledge of them due to the fact that there are many different forms of DDoS attacks. Some research, such as [43], [44], and [45], has established categories for different types of DDoS attacks.

The suggested research, on the other hand, offers a taxonomy of the three DDoS attacking strategies that are used the most frequently: volumetric attacks, protocol-exploitation attacks, and application-layer attacks. Attacks that flood TCP and UDP connections are classified as volumetric those, while attacks that flood DNS and HTTP connections are considered application-layer attacks [2].

2.4.1.1. Volume Based Attacks (Volumetric attacks)

In a volumetric-based attack, the victim is overwhelmed with massive traffic to drain its bandwidth. The magnitude of this type of attack is measured in Byte per second (Bps). However, these attacks include two main types: Flooding attacks and Amplification attacks.

2.4.1.1.1. Flood Attack

This assault is frequently carried out by sending a significant traffic volume to a target system, utilizing numerous bots on a botnet. The goal is to flood the targeted system's network capacity with this large amount of IP traffic, causing the victim host to slow down or crash, making it unable to reply to legal requests. User Datagram Protocol (UDP) flood and Internet Control Message Protocol (ICMP) flood attacks are examples of these attacks.

2.4.1.1.2. UDP Flood

This attack starts on a remote target by rapidly sending many UDP packets to random ports on the victim machine [45]. As a result, the available network bandwidth is exhausted [47]. Whenever a specific target system receives a UDP packet, the system identifies the service accepting connections on the specified port. However, suppose the system does not have a service accepting connections on that port. In that case, the server generates an ICMP packet with the message “destination unreachable” and sends it to the source address, which is usually forged. Because the targeted system's resources are restricted, delivering enough packets will prevent the victim system from responding to legitimate requests or potentially stop working altogether. Typically, an attacker spoofs the assaulting packets' source IP addresses so that the actual attackers are masked, and the zombies, who are frequently weak clients, do not get responses that might impede their network resources [48].

2.4.1.1.3. ICMP (Ping) Flood

When zombies flood the victim system with ICMP_ECHO_REQUEST (ping) packets, it's known as a DDoS ICMP flood attack. The aim is to overload the victim's incoming and outgoing bandwidth by forcing it to reply to all ping queries. Because the victim's servers frequently attempt to answer with ICMP_ECHO_REPLY packets, the overall system will be significantly slowed. It also cannot respond to a legitimate request.

2.4.1.2. Amplification attacks

This attack aims to magnify and reflect the attack by utilizing the broadcast IP address attribute on most routers, as shown in Figure 2.7.

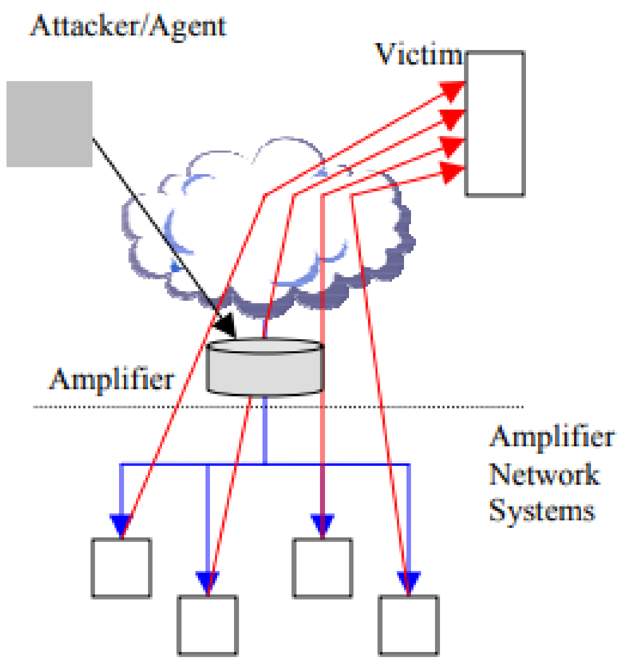


Figure 2.6. Amplification attack.

Furthermore, an amplification attack is any attack in which the attacker may double their power using an amplification factor. Amplification attacks are “asymmetric,” which means that an attacker only needs a small number or low level of resources to generate a higher level of malfunction or failure of target resources. Smurf Attack (ICMP amplification) is an example of an amplification attack.

2.4.1.2.1. Smurf Attack

Smurf malware creates faked ICMP packets with the target's IP address as the source IP address and sends these ping packets to a network amplifier (a system that provides broadcast addressing). As a result, all the network's connected hosts send ICMP ECHO REPLY packets to the victim machine. Because the attack factor is doubled by the number of active hosts in the network, and because there are so many attacking agents, the bandwidth of the target host is filled, and the host might become overloaded, unable to reply to legitimate requests [49].

2.4.1.3. Protocol Exploit Attack

The attack aims to exploit the unique features of the design or implementation of the protocol running on the target system to trick the victim into exhausting their resources [48]. The number of packets per second (Pps) is a measurement unit. TCP SYN attack is an example of a protocol attack.

2.4.1.3.1. TCP SYN Attacks

The perpetrator of a Distributed Denial of Service attack known as a TCP SYN DDoS instructs zombies to send false TCP SYN requests to a victim server in an effort to exhaust the target server's available CPU resources and stop it from responding to real requests. The TCP SYN attack takes advantage of the three-way handshake that occurs between the transmitting and receiving systems in order to overwhelm the target system with TCP SYN packets that have forged source IP addresses. Therefore, in order to react to the uninvited system, the victim's system sends an ACK followed by a SYN. The server will eventually run out of available CPU and memory resources if it handles a large number of SYN requests but does not get an ACK + SYN answer in response to those requests. If a significant number of TCP SYN attack requests are sent and the attacks are allowed to continue for an extended period of time, the victim's system will ultimately be unable to reply to legitimate users.

2.4.1.4. Application Layer Attack

These are low-volumetric attacks that make use of vulnerabilities in layer seven protocols such as HTTP. Application attacks are the most difficult to detect and counteract among DDoS attacks [50]. These are the most sophisticated and covert attacks [49] since they may be highly successful with a single machine sending low-volume traffic and crashing the web server. This makes standard flow-based monitoring technologies like IDS (intrusion detection system) exceedingly tricky to proactively monitor these attacks. Requests per second (Rps) are used to measure the Magnitude of such attacks. However, Hypertext Transfer Protocol (HTTP) flooding is one example of such an attack.

2.4.1.4.1. HTTP flood

In such an attack, distributed clients are coordinated to submit several HTTP GET requests for diverse content from a specified server, such as files, photos, or other online resources. To attack a server, using spoofed addresses or sending large amounts of data is unnecessary. Simple HTTP GET and POST requests are issued, which demand a massive quantity of data in response, wasting a lot of bandwidth and bringing the server down. The attack is most effective when the server or application is forced to allocate the maximum number of resources feasible in response to every single request.

Every one of these attacks has one thing in common: they flood the victim's network with traffic, which depletes the available resources on that network. A better comprehension of the effect that these high-volume flows have on the architecture of SDN will be facilitated by the identification and avoidance of these flows in the conventional networks that are in use today.

The essence of the attack, as well as its many forms, were briefly covered in the sections that came before this one. This study is primarily focused on the effects that DDoS has on SDN, hence a more comprehensive analysis would go beyond the scope of this research. As a direct consequence of this, the next part will concentrate on DDoS in SDN. DDoS assaults, which are becoming more common, represent a serious danger for all networks, but particularly SDN. Due to the fact that the SDN is a new design and is through the maturation process, there is the possibility that flaws in the SDN might be discovered, which can lead to a more robust protection against assaults of this kind.

2.5. DDOS ATTACKS IN SDN AND THEIR EFFECT ON CONTROLLER

The switches in SDN had no control over incoming packets and do not waste time processing them except if a rule in their tables contained instructions. So, this indicates that the controller is the first and most affected component in an attack.

Moreover, to determine the existence of the DDoS, the controller must continuously gather network traffic statistics from the switches. As a result, the controller's burden grows. Therefore, the controller's security is considered one of the most critical challenges in developing networks that depend on SDN. DDoS attacks are among the most hazardous and common in software-defined networks [36][31]. The attacker has three significant targets in attacking the SDN network: to exhaust the controller's sources, to consume the bandwidth of the channel between the switch and the controller, and to clog up the switch's flow tables with superfluous flows. When the controller attacks, the attacker uses zombie users to send many packets to the OpenFlow switch. It's hard for the controller to distinguish between malicious and legitimate traffic clearly. The OpenFlow switch checks the packet header and looks for a match with its flow table. The packet is delivered to the controller using PACKET IN message if no match is found. The controller then sends the OFPT FLOW_MOD message back. This message contains the procedure that will be performed on the packet and the timeout of the flow allocated to the packet in the flow table. And therefore, the controller's sources (memory, CPU, and bandwidth) become depleted as the number of packets routed to it increases. So, it is impossible to process new flow input for legitimate packets coming to the network, and the SDN architecture comes crashing down. Another issue arising from a DDoS attack is the overloading of switch tables. If the attack goes undiscovered, the switch tables will remain filled with flows as the controller keeps adding them. According to the OpenFlow protocol definition [51], each flow will be assigned a timeout, according to this. If the flow is idle, it will time out and be deleted from the table.

Because DDoS packets are bogus, no traffic will follow the flow is introduced, and the entry will remain in the table until it expires. The volume of incoming assault traffic, on the other hand, will be significantly faster than the volume of flows that are timed out and eliminated from the tables. In this case, the switch's tables will quickly fill up, restricting the number of additional flows that can be introduced. As a result, the network cannot process and route freshly arriving traffic correctly. Figure 2.7 depicts the discussed scenario.

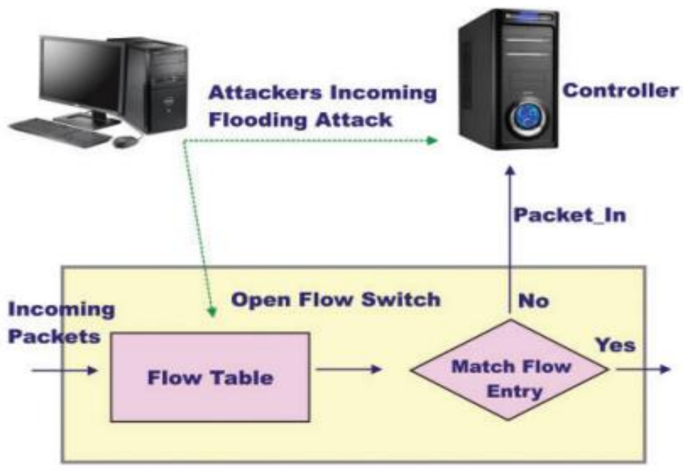


Figure 2.7. SDN controller systematic attack diagram.

2.6. CONCLUDING REMARKS

This chapter gives an introduction to SDN as well as an overview of the Open Flow protocol and its standards. A succinct explanation and classification of the several types of DDoS assaults that may be launched against SDN are also provided. As a result of reading this review, it is much simpler to identify the efforts that have been done to design a method for detecting and mitigating this attack in order to make SDN safer. The next chapter will discuss DDoS attack strategies as well as detection methods that may differentiate regular traffic from abnormal traffic.

PART 3

LITERATURE REVIEW

Comprehensive defenses against DDoS attacks have been developed in traditional network environments for decades. Several techniques for detecting such attacks and countermeasures may be used to stop them. This chapter has looked at much research done in SDN environments regarding detection and countermeasure techniques. Since the proposed study includes two crucial areas, the previous works have been classified into two main groups: DDoS attack defense techniques based on SDN and Detecting DDoS attacks with machine learning algorithms.

3.1. DDOS DEFENSE TECHNIQUES BASED ON SDN

DDoS attacks have been researched in production networks for a long time, and several strategies for detecting and mitigating them exist. On the other hand, SDN is a novel architecture that provides many functional capabilities for network management, which supports the detection and prevention of DDoS attacks[25]. This section highlights some available strategies and techniques for detecting and preventing DDoS attacks and their limitations. [52] Propose an SDN and Openflow-based intrusion detection system. Figure 3.1 depicts the architecture of such a system. The procedure renames the existing controllers to sub-controllers and adds an event processing module on top of them. As a hyper controller, the event processing module receives events from the sub-controllers and evaluates them for potential attacks. It appears that the hyper controller is intended to have a more comprehensive view of the network for a deeper understanding of the attack. However, this is only the author's conjecture. The hyper controller attack detection method is not disclosed. The solution has not been evaluated where the findings should be compared to a solution used in SDN to validate them.

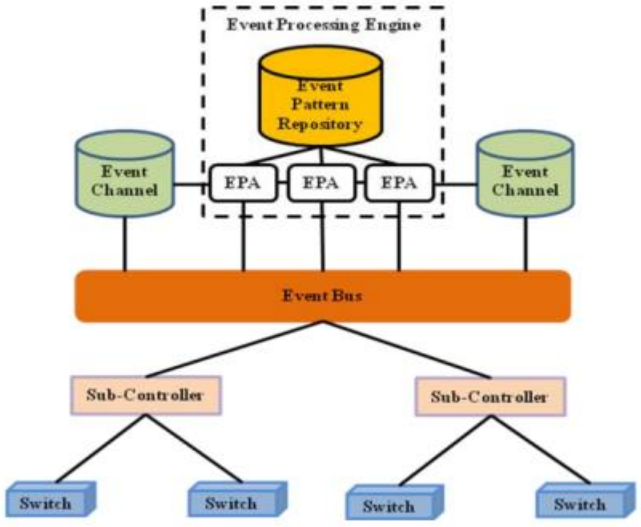


Figure 3.1. Architecture for Event-Based IDS [52].

Authors in [36] suggested an approach based on fusion entropy. This method efficiently uses the advantages of log-energy and information entropy to quickly detect the attack with an apparent change in the entropy value. The experimental results proved that the entropy decreases by 91.25% of its value in regular data traffic, which makes the method more efficient than the one that uses one type of entropy in detecting DDoS attacks. However, the threshold cannot be dynamically modified, a drawback of this method.

In [12], The author proposed a lightweight and effective solution based on the entropy variations of the destination IP to detect DDoS attacks and defend the SDN architecture of the network in its early stages. According to findings, this technique may discover the attack on a single host or a subnet within just 250 packets of malicious traffic. In addition, the threshold value was set to achieve a detection rate of 96%. However, the authors focused on the attack detection mechanism regardless of the attack's mitigation or determining its source. Also, the attack on the entire network will not be detected in this mechanism, especially since the change in the value of the entropy will then not be noticeable. On the other hand, utilizing the suggested entropy detection technique alone will lead to false positive attack detections when the network's traffic load surges during peak times with legal traffic. This is due to the given algorithm's inability to adjust to variations in traffic load dynamically. Finally, since this method is a statistical method, and these methods rely on historical network flow information, which may not accurately capture the dynamics of current network traffic due to the evolving nature of malicious network flows. Moreover, statistical techniques heavily rely on user-defined criteria that must adapt dynamically to keep pace with network changes. The computational effort required for entropy and correlation-based methods can be significant, rendering them unsuitable for real-time detection[53].

The Safe-Guard Scheme (SGS) was proposed in SDN to secure the control plane from DDoS attacks [54]. The proposed method uses DDoS attack detection behaviours that rely on the data plane and control plane cooperation. It is divided into two steps: Dynamic controller defense in the control plane, and anomalous traffic detection, which identifies fraudulent flows from authentic ones in the data plane. However, since this method relies on multiple controllers, the detection time is significantly increased. Moreover, the technique is incapable of detecting low-rate DDoS attacks.

Despite the benefits of SDN, it also comes with a variety of security risks that are inherent in traditional networks. TCP SYN flooding is one of these risks, and it is the most common of all DoS attacks, which exploit the TCP vulnerability at any form of TCP-based service or the web server end. SAFETY is a novel technique for detecting and mitigating TCP SYN flooding in the early stage, proposed in [55]. SAFETY uses the entropy approach to estimate the randomness of flow data, taking advantage of SDN features such as programming and broad visibility. The authors carried out extensive tests and found that the average response time and average assault detection time have significantly improved. However, this strategy is only suitable for dealing with a single victim, and when numerous victims are involved simultaneously, the network becomes unstable.

3.2. DETECTING DDOS ATTACKS WITH MACHINE LEARNING

In SDN, machine learning algorithms perform admirably when detecting DDoS attacks [25], [2]. The researchers in [56] employ the method called Self-organizing Maps (SOM) for detection DDoS [57]. This research demonstrates effective DDoS attack detection with high detection rates and meager false alarm rates. The accuracy of this method is between 98.6% and 99.11%. This approach trains SOM by gathering flow statistics from Openflow switches, and NOX is the controller. The metrics that are examined during SOM training include average bytes per flow, growth of single ports, an average duration per flow, growth of single flow, average packet per flow, growth of single ports, and percentage of pair flows. As time goes on and more statistics are obtained, the SOM will enhance the statistics of its vectors.

Most networks that non-SDN use this technique [25]. It follows the same process and ignores DDoS's impact on the controller when applied to SDN networks. So, before the solution can be utilized in the network, it must operate concurrently with SDN and undergo a brief training period. Another problem is that SDN could regularly alter the network. This indicates that the SOM solution must be retrained for improved security. Finally, a lightweight solution becomes a significant burden for the network as the SOMs' neurons must grow as the network grows, resulting in expansive neurons in the network. Furthermore, another issue with this approach is that false negatives will be reported when the attack parameter is set to a low value. In [58] the researchers use three machine learning methods named: Support Vector Machine (SVM), k-nearest neighbors (K-NN), Random Forest (RF), and four deep learning mechanisms such: as multilayer perceptron (MLP), convolutional neural network (CNN), Gated recurrent unit (GRU), Long short-term memory (LSTM), to detect DOS/DDOS attacks in both transport and application layers in SDN. The authors used two ways to evaluate the proposed work: 1) Using an Open Network Operating System (ONOS) controller and a Mininet network emulator on a testbed. 2) The **CICDoS2017** and **CICDDoS2019** datasets were used, and both are up to date. This work had the best results, with up to 95 % accuracy rate for application-layer DDoS attacks and up to 98 % accuracy rate for transport DDoS attacks. However, this research focused on detecting DDoS attacks without addressing the strategy of mitigating those attacks when they occur.

In [59] , the authors used machine learning algorithms such as: (RF, Decision Tree, SVM, and MLP) to detect many types of DDoS attacks named: (flow-table attacks, controller attacks, and bandwidth attacks). Scapy tools are used to make each attack. To simulate an SDN environment, Mininet with a POX controller was used. Moreover, a particular POX module was codified to gather and save the data in a data flow file to create the data set. As a result, the RF algorithm has the best accuracy, while the DT algorithm has the fastest processing time. However, online detection of DDoS attacks in real traffic wasn’t evaluated in this proposed work, nor was the mitigation or avoidance technique for DDoS attacks. In addition, the authors used the same features with the dataset to train the classifiers for detecting the three different types of attack, which caused the ‘controller attack’ to have the lowest accuracy in classification by all algorithms. So, if the researchers are more interested in this type of attack, a special classifier may be created to identify it. Other attributes might be selected to categorize this type of DDoS Attack better.

Furthermore, in [2] several feature selection methods have been employed to identify the best features from the NSL-KDD dataset for training and testing ML models. These methods can be classified into three categories: embedded, wrapper, and filter. Besides, the list of best features chosen for each feature selection technique is used as a source of information through various ML classifiers such as RF, DT, SVM, KNN, and NB (Naïve Bayes). Eventually, the experimental results demonstrate the effectiveness of the RF classifier with the recursive feature elimination (RFE), which achieves 99.97% accuracy for DDoS attack detection in SDN controllers. However, this research relied on the NSL-KDD dataset, which does not represent real-world network traffic [60]. Also, it hasn't contained any new data items, and some tuples are backdated [33]. Therefore, it was better to use the Up-to-Date Dataset.

In the SDN context, authors in [32] utilize machine learning techniques such as SVM and DT to detect SYN flooding attacks on one host in SDN. To connect the topology of the SDN network and simulate it, the Mininet simulator was used. Furthermore, the suggested model was trained and tested using the KDD99 dataset. In these conditions, experiments demonstrate that SVM outperforms Decision tree techniques. However, the ML classifiers in this work achieved low accuracy, 78% for DT and 85% for SVM. The authors in the previous study [25], [61] mainly utilized the KDD99 dataset with an SVM classifier and acquired 99% accuracy. On the other hand, The KDD'99 dataset is old [62], unbalanced, and contains duplicate data, introducing a bias into the result set [33], [25].

Table 3.1. Comparative Analysis of DDoS Attack Detection Methods Using Different Classifiers.

Method	Accuracy	Dataset	Limitations	Attack Type
SVM, K-NN, RF, MLP, CNN, GRU, LSTM [53]	Up to 95% for Application-layer DDoS attacks, up to 98% for Transport DDoS attacks	CICDoS2017 and CICDDoS2019	Focused on detection, not mitigation.	Application-layer DDoS Attacks, & Transport DDoS Attacks.
RF, Decision Tree, SVM, MLP [54]	Best accuracy with RF, fastest processing time with DT	Custom Dataset	No real traffic evaluation. Low accuracy for "controller attack".	Flow-table attacks, Controller Attacks, And Bandwidth Attacks.
RF, DT, SVM, KNN, NB [2]	99.97%	NSL-KDD dataset	Relied on outdated NSL-KDD dataset.	N/A
SVM, DT [27]	78% for DT, 85% for SVM	KDD99 dataset	Low accuracy of 78-85%. Used outdated KDD99 dataset.	SYN Flooding Attacks.
Stacking classifiers [Proposed solution]	99.99% for the Stacking model	Custom Dataset	Binary classification; multi-class approach not used.	UDP & ICMP Flood attacks, Smurf Attack, TCP SYN Attack.

Machine learning-based approaches are favored to identify suspicious activities based on strange network behavior. However, the dataset that has been used for training often affects how well these techniques [63].

3.3. CONCLUDING REMARKS

This chapter has surveyed much research performed in SDN milieus regarding detection and countermeasure techniques. Since the proposed work in this research includes two significant areas, the previous results have been categorized into two main groups: DDoS attack defense mechanisms predicated on SDN and Detecting DDoS attacks with machine learning algorithms.

PART 4

RESEARCH METHODOLOGY

This chapter presents the proposed solution for detecting and mitigating DDoS attacks on computer networks. The chapter begins by discussing the three main modules of the resolution, namely Feature Creation, Stacking Classifier, and Attack Detection. Section 4.2 outlines the necessary Environment Requirements, including the RYU Controller, Mininet Network Emulator, and Packet Generation using Hping3. The following section, 4.3, provides information about the Data Collection and Dataset, which includes data on the volume of network traffic in three categories: TCP, UDP, and ICMP. Section 4.4 describes the Feature Selection process, which utilizes a Random Forest to calculate the value of variable importance. The subsequent section, 4.5, details the Classification Methodology, which includes several algorithms, such as Random Forest (RF), K Nearest Neighbor (KNN), Decision Tree (DT), Logistic Regression (LR), Multi-Layer Perceptron (MLP), and Stacking Classifier. Section 4.6 explains the Grid search hyperparameter tuning method to optimize the solution's performance.

Machine learning aims to teach computers to carry out specific tasks without explicit instructions by learning patterns from datasets. SDN has proven helpful in identifying DDoS attacks by building predictive machine learning models. Five machine learning techniques were used on a DARPA dataset, with SVM providing the most outstanding detection accuracy and lowest false positive rate[64]. However, the dataset obtained from the simulated software-defined network environment is not compared with the analysis reported in that study.

Self-organizing Maps (SOM), an unsupervised artificial neural network, have also been utilized to detect flooding attacks in SDN [56]. The suggested method, when compared to other methods using the KDD-99 dataset, achieves a high detection rate and a low rate of false alarms in less period of computing (CPU) time by training the traffic flow with 6-tuple features.

Deep Learning is another exciting method for SDN DDoS attack detection [65][66]. Although machine learning algorithms have been used to detect DDoS in SDN, most use the KDD and NSL-KDD datasets [25], with only a few extracting real network data from SDN configuration.

This chapter focuses on detecting DDoS attacks in SDN environments, utilizing Machine Learning (ML) and MLP algorithms. The threat of DDoS attacks on network infrastructure has increased in recent years, and it is crucial to develop effective strategies for their detection and mitigation. This study employed ML/DL algorithms to detect DDoS attacks using a dataset created for this purpose.

4.1. PROPOSED SOLUTION

The proposed approach for intrusion detection, as illustrated in Figure 4.1, aims to enhance the security of SDN-based networks by employing automated and intelligent analysis of network flows, followed by intrusion detection mechanisms. This end-to-end intrusion process is based on three main modules, Feature Creation, Stacking Classifier, and Attack Detection, implemented in an SDN application at the application layer. The Feature Creation module collects network flows from the switches at regular intervals and computes the required features for each flow, which the Stacking Classifier subsequently uses. The stacking classifier applies its pre-built intrusion detection model on the flow instance and forwards the output to the Attack Detection module. Finally, the Attack detection module determines the appropriate course of action based on the classification result and installs flow rules into the corresponding switches, identifying the affected host.

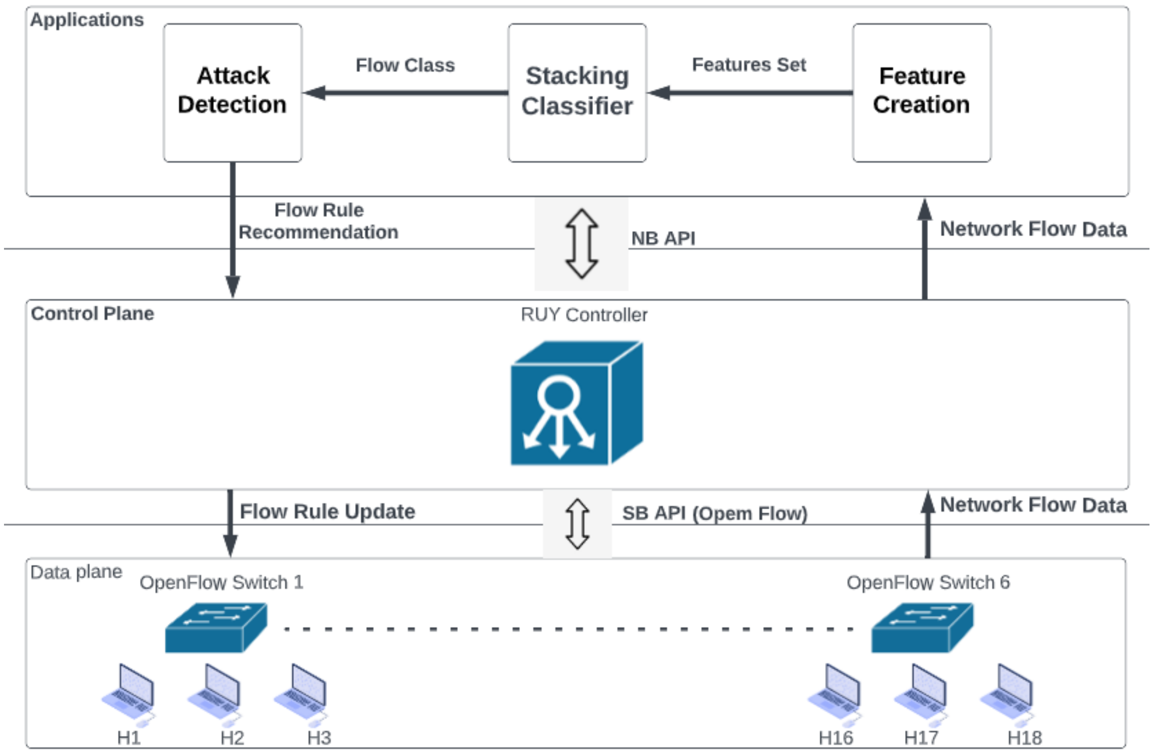


Figure 4.1. Network security with AI-Powered SDN architecture.

The process outlined involves the periodic collection of network flow entries from the switches, with features being created for every flow retrieved by the Feature Creation module. The "_monitor" function initiates the process by sending a request for statistics to all the switches registered with the controller. Once the switches receive this request, they send the relevant statistical data back to the controller. The controller then processes this data using the "_flow_stats_reply_handler" function to extract the relevant information.

After the data have been retrieved, they are placed in a file known as "PredictFlowStatsfile.csv," which is a text document that details the flow of network traffic in the network that is being monitored. After then, the "flow predict" function is invoked, which makes use of a Stacking classifier that has been pre-trained to make a prediction about the flow of network traffic based on the statistical data that has been acquired. The detected attack type as well as the source identifiers of the flow, which include the source IP address, the source MAC address, and the physical switch port the packet is originating from, are transmitted to the Attack detection module if the classification conclusion is any form of attack. The machine learning model is constantly updated by inserting fresh training data for previously identified kinds of attacks, as well as adding previously unknown types of attacks when they are found. The training data for the stacking model consisted of a variety of different DDoS attack types as well as regular traffic. This led to the formation of a binary classification model.

To summarize, this process involves collecting network statistics data from switches, processing the data to extract relevant information, storing the information in a file, and then using a pre-trained Stacking Classifier to predict the flow of network traffic based on the collected statistics.

Moreover, Figure 4.2 displays the systematic workflow of the model under consideration, which commences with acquiring data from the source. The data is then transformed into a structured dataset, which undergoes a preprocessing stage to enhance its quality and relevance. The model is then generated and applied to the preprocessed dataset to extract meaningful insights. To facilitate a comprehensive understanding of the model, each of its constituent steps is explained in detail in the subsequent sections.

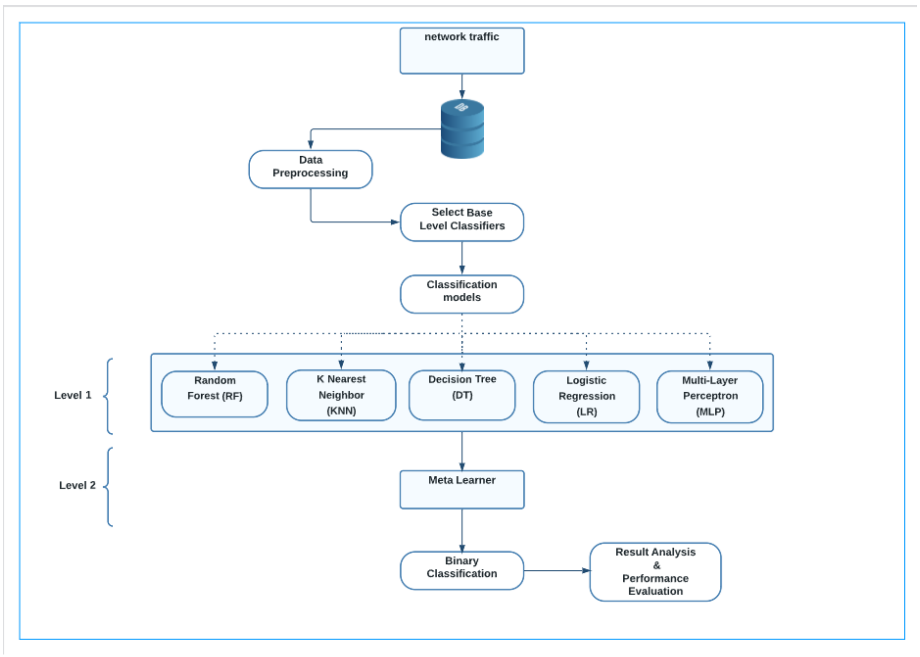


Figure 4.2. Workflow of the model.

4.2. ENVIRONMENT REQUIREMENTS

A customized topology is designed for this study to capture actual SDN data. The proposed methodology employs a Stacking classifier that integrates four machine learning algorithms and an MLP as a deep learning technique to classify DDoS attacks based on the collected data as input. The simulation environment utilized for this study is the Mininet virtualized network environment that employs the RYU controller based on the Python programming language. In addition, Hping3 scripts [67] are utilized to generate legitimate and malicious network traffic on the hosts in the simulation.

4.2.1. Controller

For the initial stage of the experiment, the selection of a controller is necessary. There are several well-known controllers available. The one utilized in this experiment is RYU[68] based SDN framework that supports the full OpenFlow version. RYU has several advantages over other SDN controllers. It is open-source, highly customizable, and written in Python, making it easy to learn and use.

Additionally, RYU is highly scalable and can manage and control networks of any size. Besides, RYU has a robust API that allows for the development of custom applications to control the network. Finally, RYU has several features that can be used to simulate a DDOS attack, such as the ability to set up virtual networks and monitor and control traffic flows.

4.2.2. Network Emulator

The network emulator utilized for this experiment is ***Mininet*** [69]. It is a standard network emulation tool that may be applied to SDN. Mininet may simulate a network using the kernel namespace feature on a laptop or PC. Due to network namespace, individual processes can have their own network interfaces, ARP tables, and routing tables. Mininet takes advantage of this kernel ability. To operate switches and hosts on the kernel, it utilizes process-based virtualization. A variety of large networks1 with various topologies may be tested and emulated with the help of a controller. In addition, the Mininet emulation-created code may be transferred to a real production network. As shown in Figure 12, Mininet makes it simple to design complex and unique network topologies that operate with several concurrent users independently.

The test beds may be created and tested with help from the Command Line Interface (CLI). Mininet also includes the ***MiniEdit*** tool, which aids in network design and implementation after user interface editing is done. One OS can support up to 4096 hosts, according to claim Mininet.

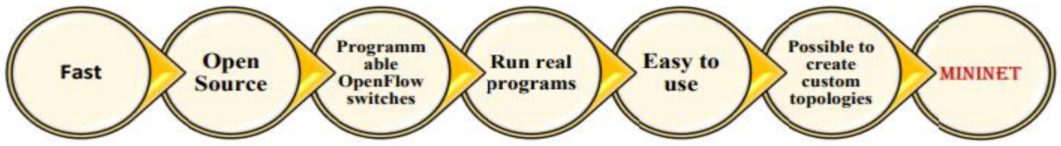

Fast
Open Source
Programmable OpenFlow switches
Run real programs
Easy to use
Possible to create custom topologies
MININET

Figure 4.3. Mininet Features.

4.2.3. Packet Generation

Hping3 [67] is used in this research to create legal and attacking traffic. *hping3* can generate ICMP/UDP/TCP packets and show target responses, even as ping does for ICMP replies.

Python is utilized to write the code that generates random source IP addresses and host IP addresses. The " randrange " function derives from the "random" function. IP addresses in Mininet are gradually assigned starting from 10.0.0.1. The source IP address for normal traffic is constructed using the range specified in the script (e.g., 10.0.0.1 – 10.0.0.19).

The random function "randrange (1,19)" establishes the destination IP addresses. The attack scripts are employed to simulate attacks on a single victim. All packets in the single-victim attack script are forwarded to the victim's destination address, specified when the attack script is called. However, the same method used for normal traffic produces the source IP addresses of attack traffic.

4.2.4. System Architecture and Setup

The experiment was performed on a Dell laptop with an Intel Core i7-10870H, 2.21GHz CPU, and 24 GB RAM as a host machine. Python is the language used to create all simulation software besides the Ubuntu server as the RYU controller's primary operating system. The operating system is Linux Ubuntu 14.04.6 and Mininet version 2.2.2, which supports the 1.4 Openflow version and is used to simulate the network's topology. Oracle Virtual Box served as the virtual environment for simulation. Six switches and 18 hosts were used to establish a network using Mininet, as shown in Figure 4.4. Network switches were Open Virtual Switch (OVS)[70]. An OVS switch utilizes both hardware and software to function. There is no distinction between Openflow and OVS switch for this work. Both do the same task, and Mininet supports both.

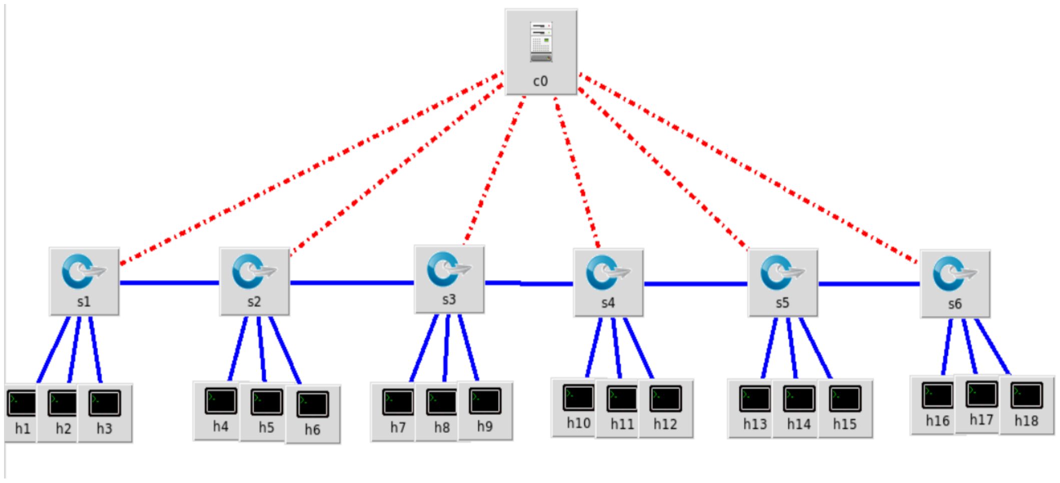


Figure 4.4. Network experiment using six switches and 18 hosts.

Even though it isn't a large-scale topology, an attacker may use packet manipulation tools on a single machine to produce spoofed packet streams. Because these streams appear to flow from several IP addresses, this environment effectively simulates a much broader topology. Additionally, the method won't change significantly depending on whether the dataset is obtained using one controller or multiple controllers [31].

Table 4.1. Experiment Setup.

Equipment/Software	Specifications
Host Machine	Dell laptop with Intel Core i7-10870H, 2.21GHz CPU, and 24 GB RAM
Virtual Environment	Oracle Virtual Box
Guest Operating System	Linux Ubuntu 14.04.6
Network Simulator	Mininet version 2.2.2
Controller	RYU controller
Switch Type	Open Virtual Switch (OVS)
Number Of Switches	6
Number Of Hosts	18
Programming Language	Python
Attack Simulation	Single machine with packet manipulation tools" ***Hping3.***"
Dataset Collection	One controller

The following subsection discusses details on the dataset produced for SDN-based network simulation.

4.3. DATA COLLECTION AND DATASET

As technology develops and the amount of data generated increases, it is increasingly important to have an accurate means to represent data before attempting to create an algorithm that can effectively solve the problem. Over time, the focus has shifted to developing algorithms that can effectively address the challenges posed by the emergence of 5G, IoT [71], and cyber-physical processes [72].

Likewise, with the rise of SDN technology, the threat of DDoS attacks has become increasingly prevalent. To better understand and protect against these attacks, researchers have developed datasets of SDN-based DDoS attack scenarios. These datasets provide a comprehensive overview of the various types of DDoS attacks that can be launched against SDN networks and the techniques used to detect and mitigate them.

The normal datasets are used as the baseline for the machine learning method to detect anomalies, and deviations from these baselines are marked as attacks [73]. Any machine learning algorithm's efficacy is determined by its ability to identify abnormalities. This necessitates a comprehensive dataset with both normal and abnormal network traffic. As a result, it is vital to ensure data accuracy and integrity to construct a reliable detection system. However, developing accurate detection algorithms is difficult due to the benchmark DDoS attack dataset that is currently available. For example, The KDD'99 dataset has many duplicate records, which frequently causes a bias in the detection results in favor of the frequently occurring forms in the training set [33].

Additionally, the researcher [74]pointed up a few issues and weaknesses in the CICIDS 2017 data. The dataset includes 203 missing information instances and 288602 missing class labels. The CICIDS 2017 dataset also appears unsuitable for any IDS training due to its enormous size and abundance of duplicated information. Furthermore, most proposals did not take advantage of the new security datasets [75][45], [58], making it impossible to evaluate the most recent DoS/DDoS attacks [25]. Moreover, these datasets illustrate intrusions that can occur in a single SDN network component without showing attack mechanisms across several SDN layers.
Consequently, a new dataset with modern DDoS attacks in SDN was gathered and analyzed to overcome the drawback of the benchmark datasets currently available, such as (1) Lack of a contemporary attack style in SDN. (2) absence of contemporary normal traffic scenarios. (3) varying how training and testing sets are distributed[76]. This means that some attack types available in the testing set of the existing benchmark data sets are not present in the training set.

The dataset used in this research consists of four different kinds of DDoS attacks, as well as realistic normal traffic activities taken from the controller every 10 seconds. Figure 4.5 briefly describes the data-cleaning procedure used to create the dataset for classification.

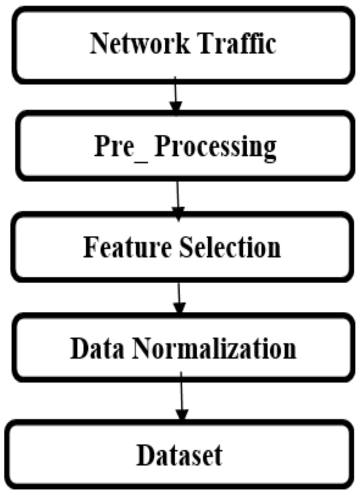


Figure 4.5. Procedures for collecting and cleaning data.

Dataset: Complete set of data that is organized in rows and columns.

This study uses the SDN Dataset created in the SDN environment. The data must be preprocessed before beginning the training of a machine learning model. However, This Dataset contains 21 features and 2,667,523 traffic flows in all. The normal data brings a total of 906853 which is labeled as "0". There are also 1760670 instances in the attack traffic with a "1" label.

Meanwhile, the dataset provided includes information on the volume of network traffic in three categories: TCP, UDP, and ICMP. The distribution of traffic by the protocol is represented as a pie graph, shown in Figure 4.6. Each section of the chart corresponds to one of the three protocols, and the size of each area is proportional to the traffic volume for that protocol.

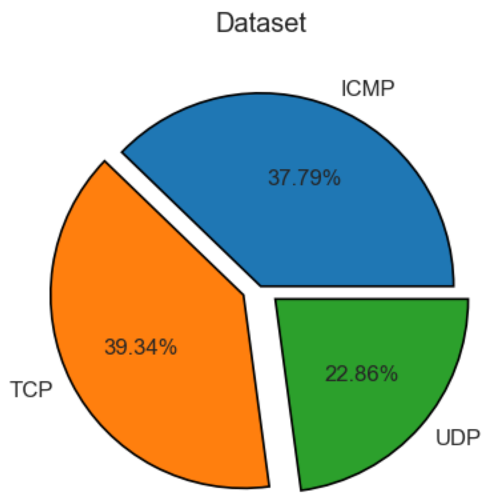


Figure 4.6. Proportional distribution of network traffic by protocol.

More profoundly, the provided dataset contains information on the volume of network traffic in six distinct categories: TCP normal, TCP DDoS, UDP normal, UDP DDoS, ICMP normal, and ICMP DDoS, which are present in Figure 4.7. The dataset consists of the number of network traffic instances in each category, which is presented as follows: TCP normal (359530), TCP DDoS (689937), UDP normal (127844), UDP DDoS (482066), ICMP normal (419479), and ICMP DDoS (588667). The data analysis revealed that TCP DDoS traffic has the highest volume among all categories, followed by ICMP DDoS, UDP DDoS, TCP normal, ICMP normal, and UDP normal traffic, respectively.

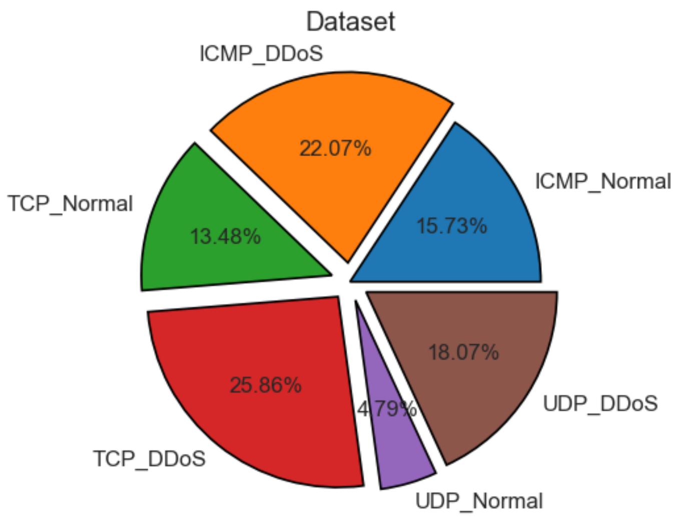


Figure 4.7. Volume of network traffic by type in a six-category dataset.

It can be noted that the dataset is considered balanced as there is no significant bias towards either normal or DDoS traffic instances. Although the amount of DDoS traffic is higher than normal traffic, the ratio of normal to DDoS traffic instances is not heavily skewed towards one category, indicating a balanced dataset. This type of balanced dataset is ideal for ML algorithms, as it ensures that the algorithm is exposed to sufficient data from normal and DDoS traffic instances.

Besides, the dataset contains various features that can be used to analyze and identify network traffic patterns. These features have been grouped into several categories, including network identifier, packet-based, bytes-based, and flow timer features. In this way, the various types of features in the dataset can be used to gain insights into the behavior and characteristics of network traffic.

1. Network identifiers feature (socket information features): identify the source and destination of network traffic. These features include IP addresses (ip_src, ip_dest), port numbers (tp_src, tp_dest), and protocol type(ip_proto) and (flow_id). it can be used to track the origin and destination of network traffic and identify malicious or suspicious activity.
2. Packet-based features: describe the characteristics of individual packets in a network—for example, packet_count.
3. Bytes-based features measure the size of data being sent or received, such as byte_count.
4. Flow timer features are attributes that measure the duration of a flow. Include:
   4.1. flow_duration_sec.
   4.2. Idle_timeout: measures when a flow can remain inactive before it is terminated. 4.3. Flow end time and flow idle time.
   4.4. Idle_time: the amount of time a flow was inactive.
   4.5. Hard_time: is the amount of time a flow was active.

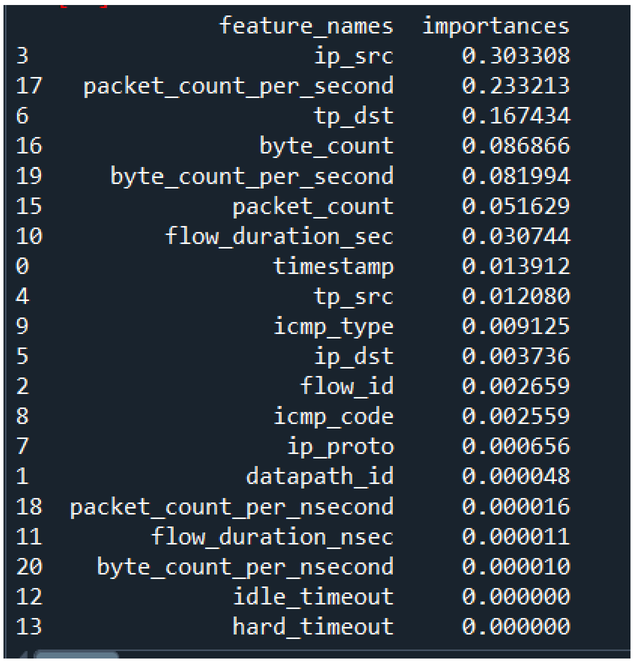
These features can be used to identify patterns and trends in network traffic and make predictions about future data. It can also be used to detect anomalies and malicious activity.

In addition, since the features have varying ranges, they need to be normalized such that the scale of values is restricted to falling somewhere between 0 and 1. Data normalization helps to guarantee that all of the characteristics included within a dataset are measured on the same scale, which can contribute to an increase in the accuracy of machine learning models. In addition to this, it helps decrease the impact of outliers as well as the computational cost of training models.

Finally, the dataset was split into two files using the ratio scale measurement method to obtain accurate findings for this experiment. 75:25 is the ratio used to partition this dataset, 75% for the training set and 25% for the testing set [77]. This allows the model to be trained on a large enough dataset while still providing enough data for testing the model's performance [25]. These training and testing sets will then be saved as CSV files.

4.4. FEATURE SELECTION

When analyzing the data used to design and optimize the model, it was observed that some features had minimal contribution or even negatively impacted model accuracy when the number of insignificant features was high [78] [79]. To address this issue, random forests to calculate the value of variable importance [80] was utilized for feature selection. The first few features obtained through this method, listed in descending order of significance, are presented in Figure 4.8 below.



```
                feature_names  importances
3                      ip_src     0.303308
17   packet_count_per_second     0.233213
6                      tp_dst     0.167434
16                 byte_count     0.086866
19     byte_count_per_second     0.081994
15               packet_count     0.051629
10          flow_duration_sec     0.030744
0                   timestamp     0.013912
4                      tp_src     0.012080
9                   icmp_type     0.009125
5                      ip_dst     0.003736
2                     flow_id     0.002659
8                   icmp_code     0.002559
7                    ip_proto     0.000656
1                 datapath_id     0.000048
18  packet_count_per_nsecond     0.000016
11         flow_duration_nsec     0.000011
20    byte_count_per_nsecond     0.000010
12               idle_timeout     0.000000
13               hard_timeout     0.000000
```

Figure 4.8. Importance of the features.

The feature selection process that is supported by random forests estimates the number of votes for the proper class by utilizing the out-of-bag (OOB) cases that are present in each tree. This allows the algorithm to evaluate the relevance of a given feature. After performing a random permutation on the values of the characteristic in question, this number is then compared to the number of votes that were cast for the correct class. The raw significance score for the feature is calculated by taking the average difference between these two values and averaging it. This value is then divided by the feature's standard error to get at the z-score. After that, the significance of the feature is evaluated by subtracting this z-score from zero. This gives the value of the feature's relevance. A representation of the feature significance values, with the order of importance decreasing from highest to lowest, is shown in Figure 4.9.

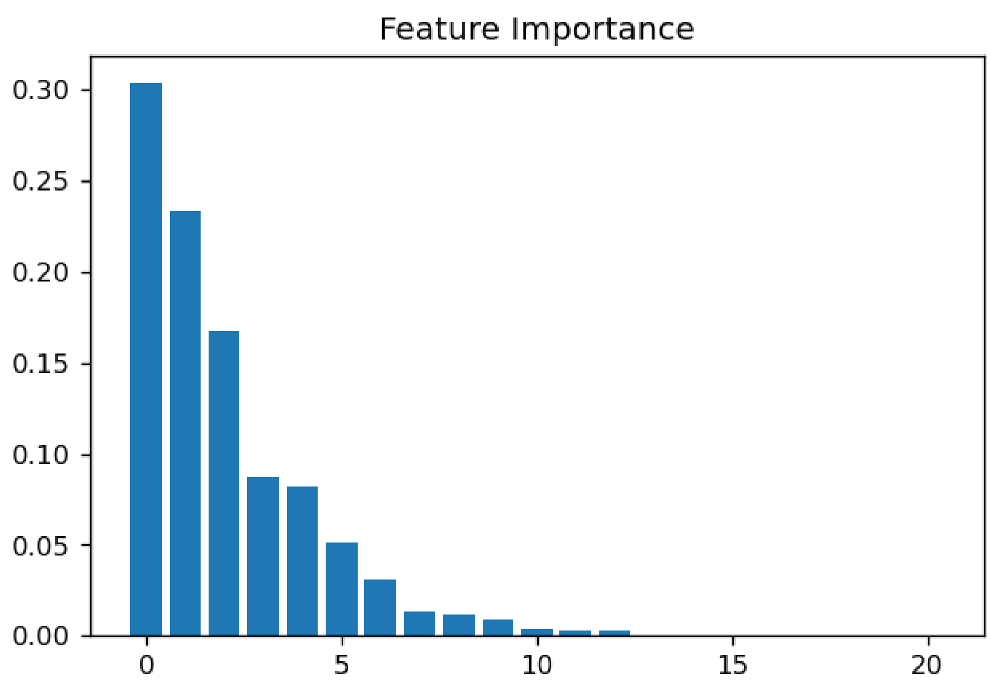


Figure 4.9. Feature importance visualization plot using RandomForestClassifier.

As depicted in the previous figure, The bars in the plot represent each feature's importance, with the higher bars representing the most critical features. The plot can be used to identify the key features that should be used to make accurate predictions. Additionally, this plot can help determine which features are unnecessary for making accurate predictions.

Accordingly, it is noticeable that the intrusions are likely related to the source of the IP address. This indicates that the origin of the IP address may be an informative and relevant factor when detecting intrusions.

However, feature selection is a critical component in this research due to the potential for overfitting when many features are used in training and testing on datasets with limited data [81]. Having too many features can lead to sampling error, resulting in a decrease in the model's ability to generalize. As such, it is important to carefully identify the most useful features that have the most impact on model accuracy. So that the feature selection was applied to remove fields such as datapath_id, idle_timeout, hard_timeout, flags… that were determined to have little impact on the investigation.

4.5. CLASSIFICATION METHODOLOGY

Machine learning algorithms can learn from data and create models to uncover hidden insights and detect anomalies without requiring explicit programming. This makes them well-suited for detecting DDoS flooding attacks by identifying deviations in traffic flow. However, selecting the most appropriate machine learning model for a given dataset is essential for accurate classification and prediction. The model's accuracy is critical in determining the quality of the predictions and providing scientific evidence for decision-making and policy formulation. This study evaluates a variety of machine learning models for their effectiveness in detecting DDoS attacks in SDN environments, and the relevant characteristics of each model will be discussed in detail.

4.5.1. Random Forest (RF)

Random Forest (RF) is a robust ML algorithm that is used for supervised learning tasks. It uses numerous decision trees to create a forest of decision trees and combines the results of each tree to make a more accurate prediction [82]. To understand how random forest works, one must first understand a decision tree. A decision tree is a structure that divides data into distinct branches based on certain conditions or thresholds. The mathematics behind this structure involves using a recursive function to construct the tree from the bottom up. This function recursively splits input data into smaller subsets until all the data points in each subset have an identical output value. Once the recursive process is complete, a decision tree is formed, which can be used to make predictions about unseen data.

The random forest algorithm combines multiple decision trees to make predictions using an assembling technique called bagging. Bagging stands for 'bootstrap aggregating,' which involves taking various samples from the dataset with replacement so that each instance contains some elements of both classes. These samples are then used to combine several decision trees to form an ensemble model. This bagging technique helps reduce variance between separate models while increasing overall accuracy.

This algorithm also relies on several mathematical operations such as entropy, information gain, and Gini impurity for training and fitting the model. Entropy measures how much uncertainty exists in each set of data points. Information gain measures how different attributes in a dataset can be used to divide data points into distinct subsets. Gini impurity measures how “pure” each subset of the data point is. Using these operations, random forest finds optimal thresholds for splitting datasets into distinct branches and building accurate prediction models.

Figure 4.10 illustrates a decision tree that was generated using a random forest in order to perform a job involving the identification of network intrusions. This perspective of the tree indicates that the entropy of the surrounding environment diminishes as one gets closer to the leaves. A particular threshold of the feature values is used by the algorithm to identify which nodes further up in the tree should be divided into two separate nodes. A random forest is a kind of machine learning model that uses a variety of decision trees, each of which is generated using a randomly chosen portion of the training dataset.

Visualizing a Sample Decision Tree from Random Forest can help gain insight into the model's inner workings. The decision tree can show which features are most important for predicting whether a network intrusion has occurred. It can also reveal any biases or inconsistencies in the data. Visualizing the tree can help identify the most valuable features and provide an opportunity to fine-tune the model’s hyperparameters [83].

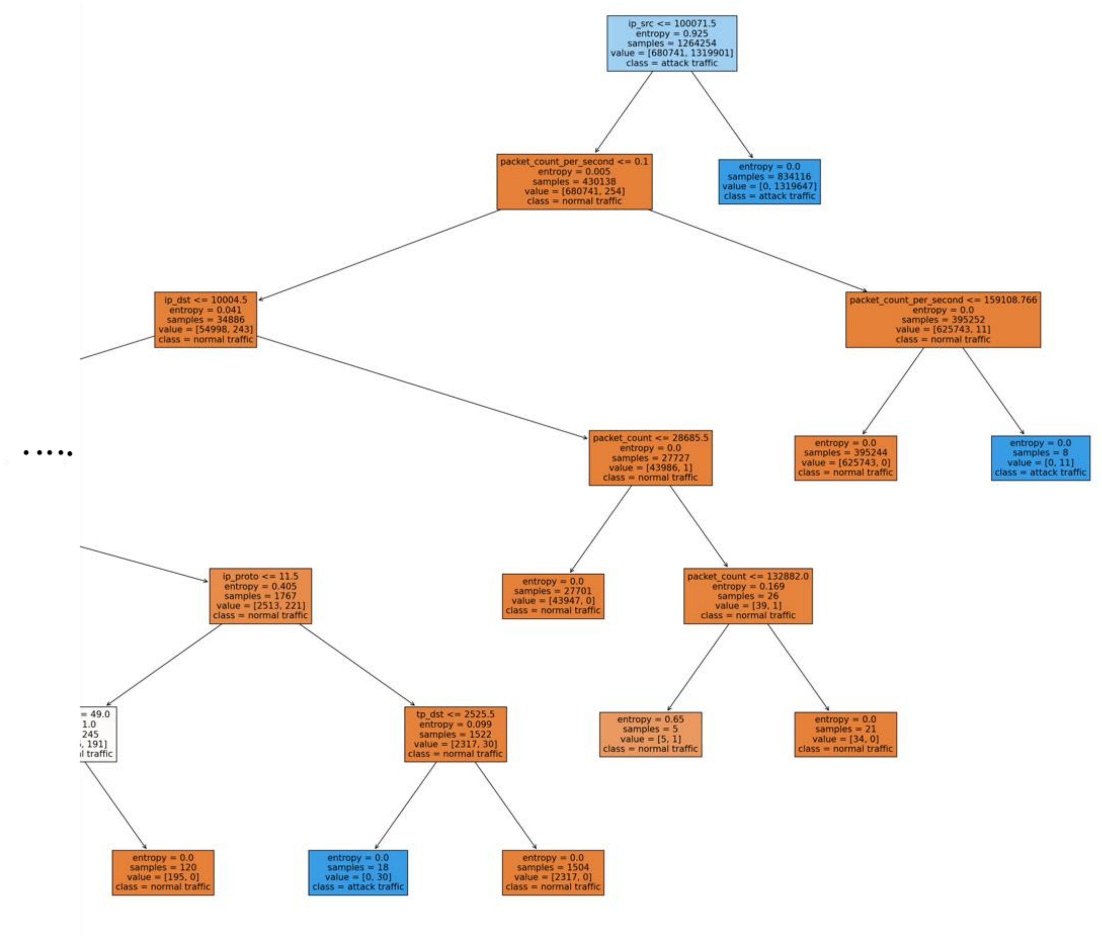


Figure 4.10. An example random forest tree for a network intrusion detection task.

4.5.2. K Nearest Neighbor

K Nearest Neighbor (KNN) is a supervised machine learning technique that makes use of a distance metric to determine which K examples in the training data are the most similar to new cases. It does this by finding the K neighbors with the shortest distance. The forecast for the new instance is then determined by the algorithm by taking the average of the results obtained by the neighbors. This technique is a non-parametric strategy that may be used for classification as well as regression work. This means that it does not make any assumptions about the distribution of the data that is being used. Because of this, it is well suited for discovering patterns in large data sets.

To guarantee precise results from the KNN prediction method, the appropriate value for k must be selected. When the value of K in the k-nearest neighbor method is increased, the algorithm's predictions become more dependable and accurate as a result of the majority voting up to a certain point without adding inaccuracy. This is because the process uses majority voting. On the other hand, when the value of K is reduced until it reaches 1, there is no longer a majority vote, which results in forecasts that are less dependable and more prone to mistakes. In addition, KNN trains sophisticated models rapidly and is resistant to noisy training data [80], which means that it is able to deliver correct results even when the data being used is not ideal.

4.5.3. Decision Tree

The Decision Tree (DT) method is a kind of supervised learning that is used to solve issues involving classification and regression. In order for it to operate, a model is first constructed that is able to forecast the value of a target variable by learning basic decision rules that are inferred from the properties of the data, and the decision tree itself also continuously developed [81]. The nodes located at the root, decision, branch, and leaf of a DT make up the majority of its constituents. Each node in the tree illustrates a choice that was made about a component of the whole data set. According to the equation, the information theory, also known as entropy, is used in order to determine the location of the root, the nodes, and the leaves , as specified in Eq. (1)

$$\text{Entropy (P)} = -\sum_{i=1}^{N} Pi \log (Pi) \tag{4.1}$$

Where p is the data's probability distribution, entropy is a measure of the randomness or uncertainty in each data set. In decision tree algorithms, entropy is used to measure the efficiency of splitting a node based on a particular feature. An entropy value near 0 indicates that the split is efficient and can be used to create a more accurate model. In contrast, a higher entropy suggests that the separation is less efficient. In network intrusion detection tasks, using entropy in the decision tree can be used to identify malicious activities quickly.

There are several hyper-parameters that have the potential to regulate DT structure. These hyper-parameters include the maximum number of splits, the minimum leaf size, and the minimum parent size. Because of this, it is very necessary to do DT optimization in order to develop a model that is straightforward and applicable.

On the other hand, in order to construct a DT, the following procedures are taken:

1. A training set and a test set are created from the data set.
2. The training set is used as the tree's root input.
3. The root is identified by considering information theory described in Equation (1).
4. The tree is pruned to have a single node at each node.
5. Continue with steps 1 through 4 until all nodes are leaf nodes.

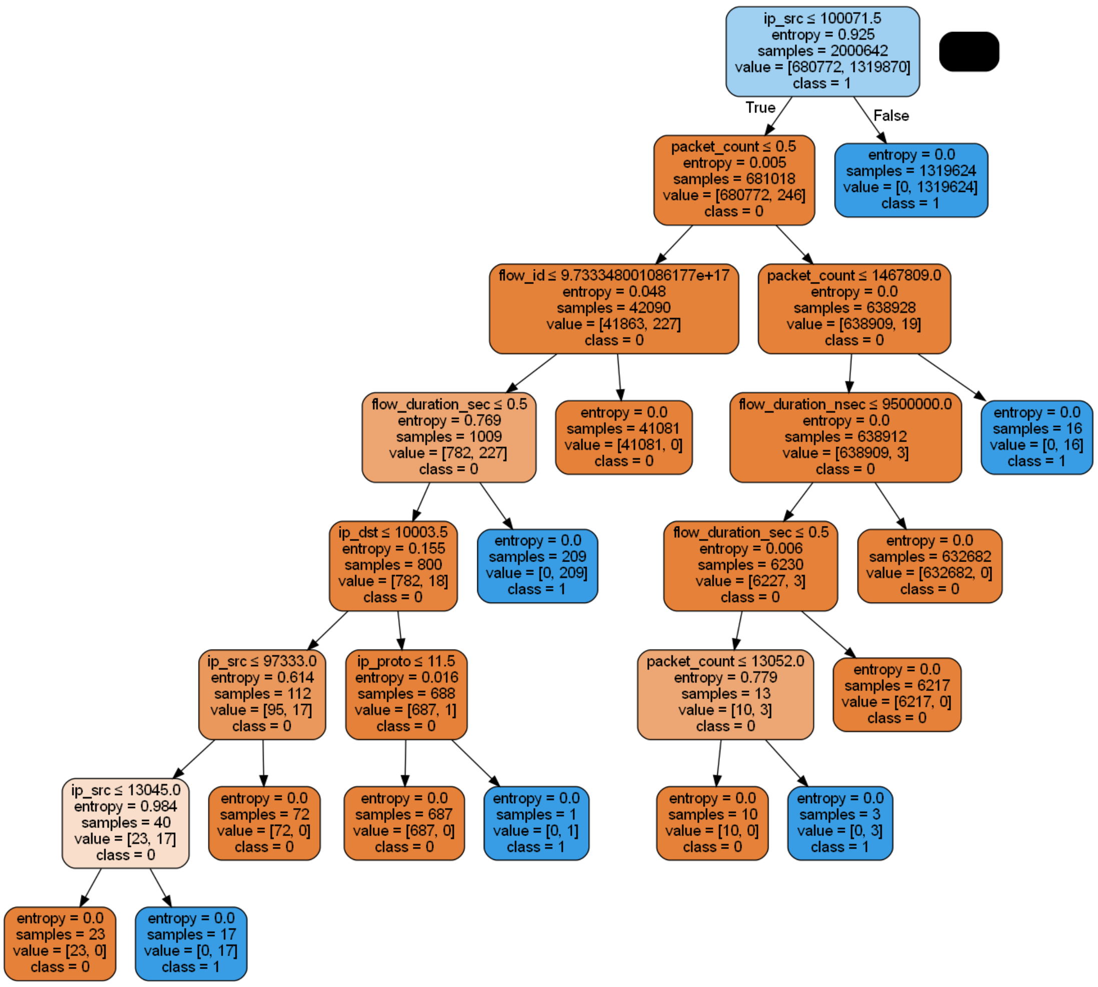


Figure 4.11. Decision tree for a network intrusion detection task

A perspective of a decision tree that was built for a sample network intrusion detection job based on the SDN dataset is shown in Figure 4.11. However, A decision tree graph is a hierarchical structure that may be used to depict the choices that are produced by an algorithm that uses decision trees. It is made up of nodes and branches, with each node standing for a choice and each branch expressing the many potential results that decision may have. This graph may be used to rapidly determine the most significant decisions that are being made by the algorithm, in addition to the connection that exists between the various options. It is a potent instrument for gaining an understanding of the algorithm's behavior and for assisting in the process of fine-tuning the algorithm's parameters in order to generate more accurate models. The entropy of the nodes decreases as they go closer to the leaves, as seen in the figure.

4.5.4. Logistic Regression

A classification model known as logistic regression (LR) is one that makes use of a mathematical function to make predictions about the likelihood of a binary event. It is a supervised learning algorithm, which means that it learns from data that has been labeled in order to generate predictions about data that has not yet been seen.
The logistic function and the sigmoid function serve as the foundation for this approach. This function takes any real-valued input and returns a number between 0 and 1 as the output. This value may be understood as the likelihood that the positive class will occur. The following is a definition of the logistic function:

$$p(x) = 1 / (1 + e^{(-z)})$$

Where:

p(x): is the predicted probability of the positive class,
x: is a vector of input features,
z: is a linear combination of the input features and their corresponding weights,
e: is the base of the natural logarithm.

The logistic regression model is capable of dealing with both continuous and categorical variables that are input. In order for categorical characteristics to be used as input to a model, they are normally one-hot encoded binary variables after going through a conversion process. LR provides a number of benefits when used to machine learning. It is a straightforward model that is easy to understand and can be rapidly trained using extensive datasets. Additionally, it is able to deal with noisy or missing data and is less prone to overfitting than other models that are more sophisticated. However, it does have a few drawbacks, such as the assumption that there is a linear connection between the input characteristics and the output probability, as well as the difficulty in dealing with extremely non-linear correlations. Nevertheless, it is still a useful tool. In machine learning, logistic regression is a model that is both practical and commonly utilized for classification problems.

4.5.5. Multi-Layer Perceptron (MLP)

In the field of artificial neural networks, a Multi-Layer Perceptron (MLP) is a form of neural network that is often used for classification and regression problems. There are many layers of neurons that make up this structure. Every neuron is a mathematical function that receives one or more inputs, performs a weighted sum on those inputs, includes a bias term, and then runs the result through an activation function.

In a multilayer perceptron (MLP), the input layer is made up of input nodes, which are responsible for receiving the characteristics and attributes of the input data. The output layer is made up of nodes known as outputs, which are responsible for producing the network's final predictions or outputs. It is possible for there to be one or more hidden layers containing hidden nodes in between the input and output layers. The hidden layers enable the network to learn complicated patterns in the input data by modifying the inputs via non-linear functions. This is made possible by the fact that the network has access to the data.

The Scikit-learn platform [84] is utilized to implement all the discussed ML techniques.

4.5.6. Stacking Classifier

Once the data has experienced preprocessing, generating the model is next. This study focuses primarily on the machine-learning technique of stacking, as depicted in Figure 4.12.

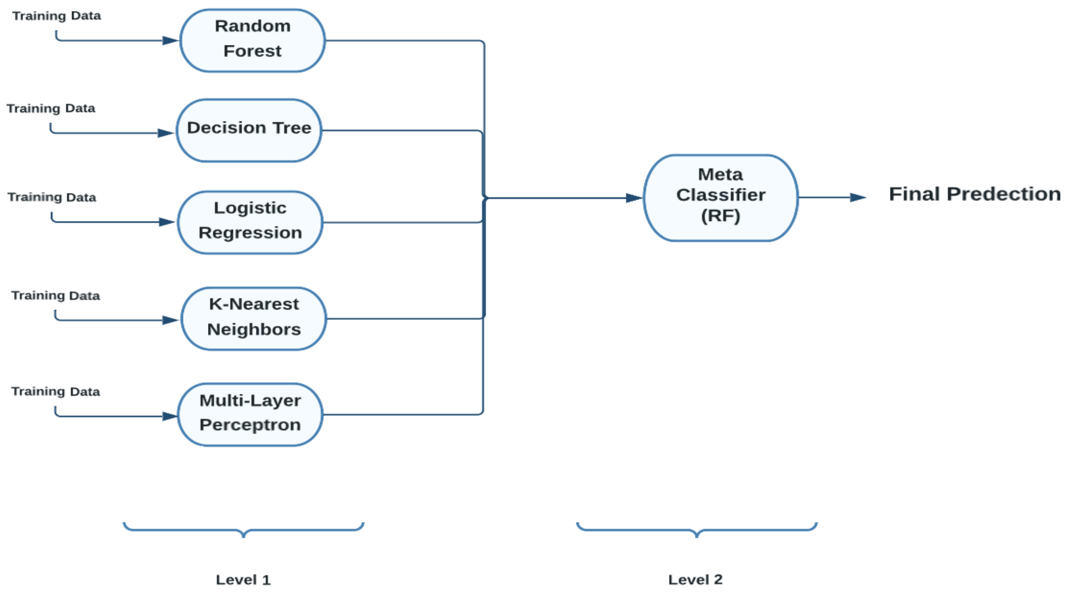
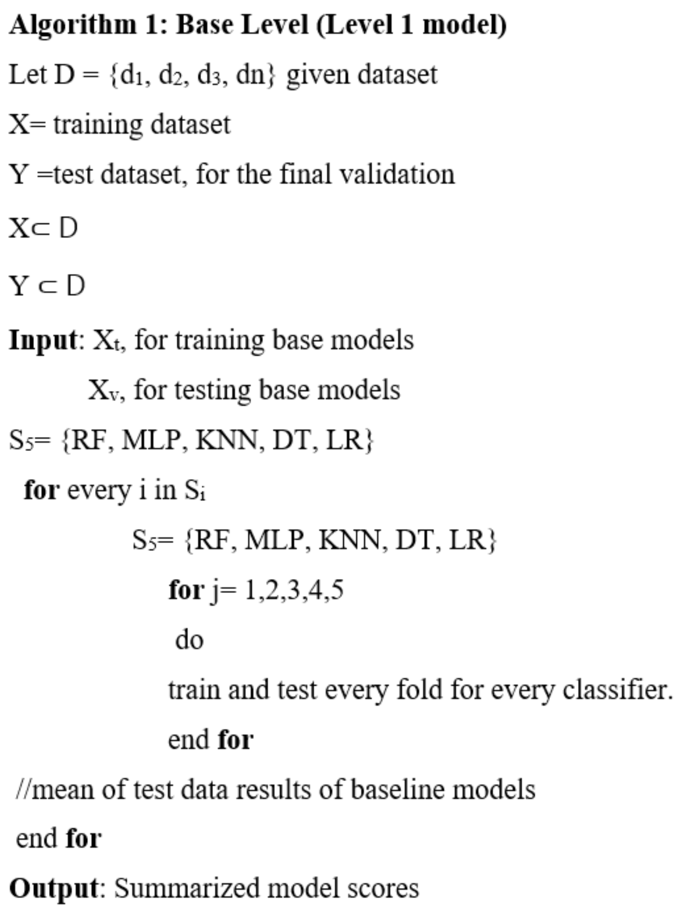


Figure 4.12. Stacking concept diagram.

A greater degree of prediction model performance may be reached by the use of the stacking methodology, which combines many tiers of classifiers. A collection of common machine learning algorithms makes up the first level, which is often referred to as the "base level" or "base learners." The information gleaned from the primary level is used by the second level, often known as the "meta-level" or "meta learners" [85][27]. The heterogeneous nature of multiple classification algorithms is leveraged by stacking them in two levels, enabling the optimal combination of the strengths of each classifier [86]. This is especially beneficial as every classifier comes with specific strengths and drawbacks. In this study, the following classifiers were employed as base learners: Random Forest (RF), Decision Tree (DT), Logistic Regression (LR), Multi-Layer Perceptron (MLP), and K-Nearest Neighbors (KNN). To ensure model performance, 5-fold cross-validation was performed on the preprocessed data to avoid overfitting. This involves dividing the data into five equal parts or folds, using four parts for training and one part for validation, and repeating the process five times, with each part used for validation once. This technique helps reduce variability and improve the model's generalization performance [87]. Finally, the average scores across all the K iterations of the K-fold cross-validation loop was calculated as shown in Figure 4.13 below.



```
Algorithm 1: Base Level (Level 1 model)
Let D = {$d_1$, $d_2$, $d_3$, dn} given dataset
X= training dataset
Y =test dataset, for the final validation
$X \subset D$
$Y \subset D$
Input: $X_t$, for training base models
       $X_v$, for testing base models
$S_5$= {RF, MLP, KNN, DT, LR}
  for every i in $S_i$
        $S_5$= {RF, MLP, KNN, DT, LR}
          for j= 1,2,3,4,5
          do
          train and test every fold for every classifier.
          end for
 //mean of test data results of baseline models
 end for
Output: Summarized model scores
```

Figure 4.13. Text text.

Several classifiers were evaluated for the meta-learner level, and the RF classifier was selected for various reasons [88]. Firstly, RF is an ensemble method that combines multiple decision trees, which can increase the diversity of the base estimators used in the stacking classifier. Secondly, it can handle numerous input features, which is beneficial when dealing with high-dimensional datasets [89]. Finally, RF is less prone to overfitting than other ML algorithms, which can improve the generalization performance of the stacking classifier. Therefore, the RF is used as the final estimator in a stacking classifier based on its ability to handle high-dimensional datasets with complex interactions, provide feature importance measures, and exhibit robustness to noise and outliers, among other advantages. It is vital to have a diverse set of base learners to produce results based on different assumptions. Therefore, five of the classifiers mentioned above were chosen.

This stacking approach allows the meta-level to determine the best way to maximize the strengths of each base learner and produce optimal prediction results.

Consequently, the stacking classifier has several advantages over single-model classifiers [90]. First, the stacking classifier can be more accurate than any base models used individually. Second, the stacking classifier can be more robust than single-model classifiers, as it can overcome the weaknesses of individual models. Finally, the stacking classifier can be used to combine models with different learning characteristics, improving the overall prediction's generalization performance [91]. Stacking is a powerful and flexible ensemble learning method that can be applied to a wide range of classification problems and significantly improve machine learning models' predictive performance.

4.6. HYPERPARAMETER TUNING

The process of determining the ideal settings for the machine learning model's hyperparameters, which may have a considerable influence on the overall performance of the model, is referred to as hyperparameter tuning. This may be accomplished by testing out a variety of various configurations of hyperparameters and picking the one that provides the most beneficial outcomes. However, since adjusting a large number of hyperparameters may be a computationally intensive process, it might be difficult to identify the hyperparameter settings that are best [89]. Grid search [90] is a technique that enables hyperparameter tuning to be carried out in a methodical manner by methodically reviewing each set of hyperparameter values automatically throughout the process of model training [92]. This method looks through all of the potential combinations of hyperparameters by requiring the user to define a range of possible values for each hyperparameter. It then chooses the combination of hyperparameters that results in the highest level of performance.

Table 4.2 enumerates the parameters that were specified for the classifiers under consideration. The values of the parameters were pre-determined by the sci-kit learn library.

Table 4.2. Parameters specified for various classifiers.

Parameters	Classifiers
Random Forest	n_estimators=25, criterion="gini", max_depth=1
Decision Tree	criterion='gini', random_state=10, splitter= 'random', min_samples_split=10, max_depth=6
Logistic Regression	(max_iter=500
Multi-Layer Perceptron	activation='tanh', solver='lbfgs', learning_rate='constant', alpha=0.0001, hidden_layer_sizes=(100,50,3), random_state=33)
K-Nearest Neighbors	n_neighbors=50, metric='minkowski', p=2

One of the advantages of grid search is that it allows for independent hyperparameter settings, making it suitable for parallel computing [93]. This means that the tuning process can be distributed across multiple computing nodes, reducing the computation time required to find the optimal hyperparameters.

The implementation of grid search was done in Python using the scikit-learn package, which offers a robust set of tools for machine learning tasks [94], [95], [45] Grid search involves assigning a series of values to each hyperparameter. The program iterates through all possible combinations of hyperparameter values to train and evaluate the performance of different models. Each variety of hyperparameter values is referred to as a hyperparameter setting. Multiple rounds of grid search were conducted, assigning a distinct set of values to each hyperparameter, to identify the optimal combination of hyperparameters that yielded the best model performance.

4.7. CONCLUDING REMARKS

In conclusion, the proposed solution presented in this chapter offers a comprehensive approach to detecting and preventing network attacks. The methodology involved using three main modules: Feature Creation, Stacking Classifier, and Attack Detection, along with various environmental requirements such as RYU Controller and *Mininet*

Network Emulator. The data collection process and feature selection using Random Forests were crucial in ensuring the accuracy of the classification methodology, which included Random Forest, K Nearest Neighbor, Decision Tree, Logistic Regression, Multi-Layer Perceptron, and Stacking Classifier. Finally, implementing hyperparameter tuning using the Grid search method improved the solution's overall performance. This methodology provides a solid foundation for addressing network security concerns in today's digital landscape.

PART 5

EXPERIMENTAL EVALUATION

Model evaluation is a critical step in experimental evaluation, where the performance of the trained machine learning (ML) model is assessed using various performance metrics. In the context of ML-based DDoS detection techniques in an SDN environment, choosing performance metrics is crucial to understanding the strengths and limitations of the different ML models and selecting the most effective ones for deployment in real-world scenarios.

This chapter presents the experimental evaluation of the proposed machine learning-based DDoS detection system. Section 5.1 provides a detailed analysis of the confusion matrix that is obtained by evaluating the system's performance. In Section 5.2, the results of various classifier models, including Random Forest, K Nearest Neighbor, Decision Tree, Logistic Regression, Multi-Layer Perceptron (MLP), and Stacking Classifier, are presented and compared. Section 5.3 discusses the validation and evaluation process of these models. Furthermore, Section 5.4 introduces the Receiver Operating Characteristic (ROC) Curve, which is used to evaluate and compare the performance of different models.

5.1. CONFUSION MATRIX (CM)

A confusion matrix (CM) is a table that is used in the process of evaluating the effectiveness of a machine-learning model. It shows the proportion of accurate and inaccurate forecasts produced by the model in relation to the results that were obtained (i.e., ground truth).

The CM is structured in the form of four quadrants, each of which represents one of the four potential outcomes of a binary classification issue. These possibilities are as follows: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for a given set of predictions [96].

In the context of an issue involving binary classification, the terms "correct predictions" and "number of correct predictions" refer, respectively, to the number of positive and negative classes. On the other hand, FP and FN stand for the total number of erroneous predictions made for the positive and negative classes, respectively. Having said that, this matrix is often displayed in the form of a table that has two rows and two columns.

The classes that were really present are represented by the rows, whereas the classes that were expected to be present are represented by the columns. Each field in the table has a number that reflects the total number of occurrences that may be assigned to a certain category. A problem with binary classification may have one of four alternative results, which are represented by the four cells in the table.

The number of occurrences that were accurately identified as positive is shown in the true positives (TP) column. The number of occurrences that have been accurately labeled as negative is shown in the true negatives (TN) column. The number of cases that were wrongly labeled as positive is shown in the false positives (FP) cell. The number of cases that were mistakenly categorized as negative is shown in the "false negatives" (FN) field of the table.

Calculating several performance measures for the classification model, such as accuracy, precision, recall, and F1 score, may be accomplished with the assistance of the data that is provided in the confusion matrix. These metrics provide a glimpse into the model's strengths and limitations, and they may be put to use either to fine-tune the model or to evaluate it in comparison to other models.

In the context of detecting DDoS attacks, a confusion matrix can be used to evaluate the performance of the detection method in terms of its ability to correctly identify DDoS attacks and differentiate them from legitimate traffic. This evaluation can be done in terms of the method's ability to identify and distinguish DDoS attacks from legitimate traffic. The performance indicators that are obtained from the confusion matrix may give useful insights into the strengths and limits of the detection approach, as well as assist identify areas in which improvements can be made.

However, it is essential to note that a confusion matrix alone may not comprehensively evaluate the DDoS attack detection method's performance. Other evaluation methods may be necessary to provide a more complete evaluation.

Moreover, the confusion matrix allows calculating several performance metrics for a binary classification model, including:

1. Accuracy: The proportion of correct predictions the model makes, calculated as (TP + TN) / (TP + TN + FP + FN).
2. Precision: The proportion of true positive predictions out of all positive predictions made by the model, calculated as TP / (TP + FP).
3. Recall (or sensitivity): The proportion of true positive predictions out of all actual positive instances, calculated as TP / (TP + FN).
4. F1 score: The harmonic means of precision and recall, calculated as 2 * (precision * recall) / (precision + recall)).

It is important to note that selecting performance metrics that align with the application's goals and requirements. Precision and recall are vital metrics that can be used to assess the effectiveness of ML-based DDoS detection techniques[97]. Precision is an excellent metric for minimizing false positives, while recall is useful for reducing false negatives.

The F1 score is a helpful metric as it balances precision and recall equally and can be chosen when there's a trade-off between the two or when both metrics are essential [97]. The selection of performance metrics is critical for evaluating the efficacy of ML-based DDoS detection techniques. The F1 score is a good option for balancing precision and recall.

5.2. CLASSIFIER MODELS RESULT

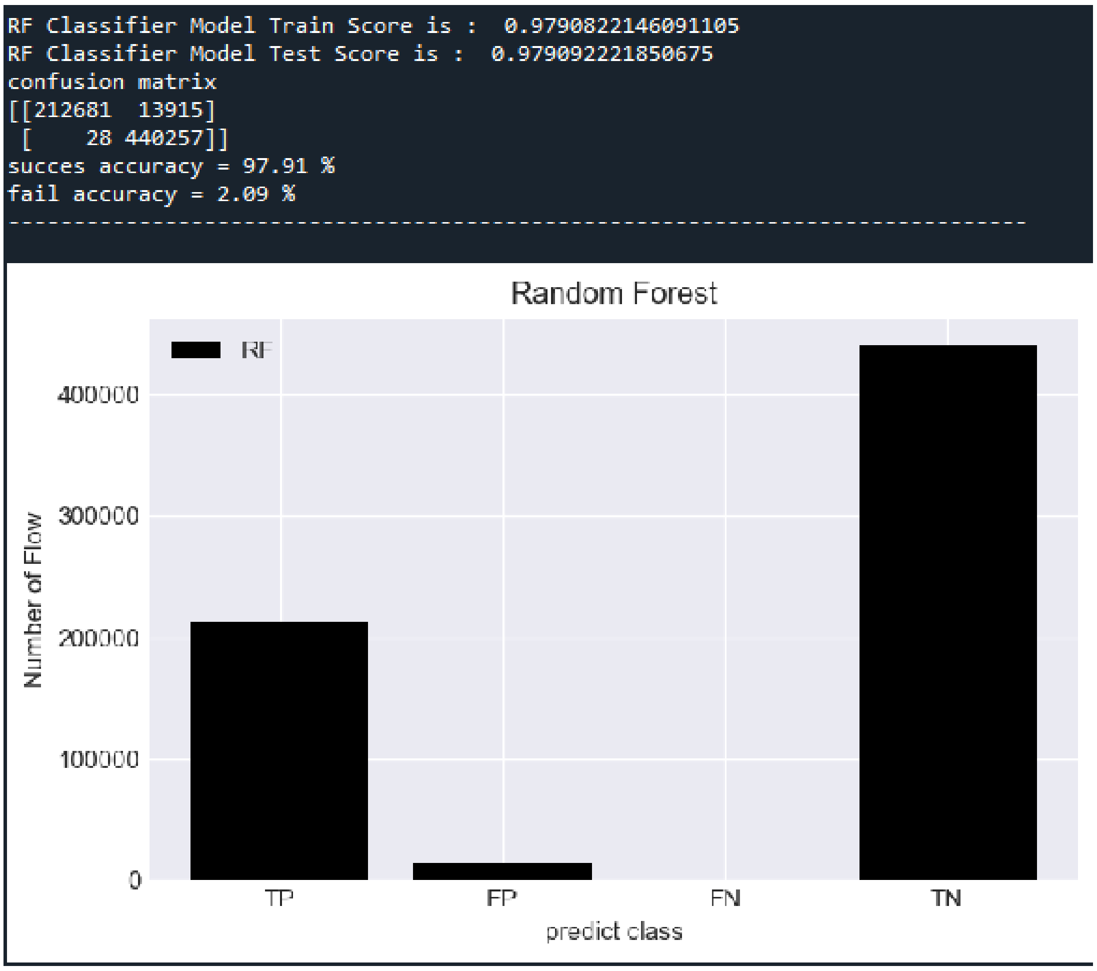
This section presents the performance of several machine learning classifiers on the dataset proposed in Chapter 4. A 75:25 train-test split was performed using the ratio scale splitting method to ensure a fair and unbiased evaluation of the models.
Five common supervised learning algorithms were evaluated:

5.2.1. Random Forest

The Random Forest Classifier model was trained and evaluated on the proposed dataset. The model's training accuracy score was found to be 0.9790822146091105, indicating that the model could correctly predict 97.91% of the instances in the training data. Similarly, the model's test accuracy score was found to be 0.979092221850675, indicating that the model could correctly predict 97.91% of the instances in the test data.

The confusion matrix was also calculated to better understand the model's performance. The confusion matrix shows that out of 212681 + 13915 = 226596 actual negative instances, the model correctly predicted 212681 instances and incorrectly predicted 13915 instances as positive. Similarly, out of 28 + 440257 = 440285 actual positive instances, the model correctly predicted 440257 instances and incorrectly predicted 28 instances as negative.

Finally, the success and fail accuracies were calculated to analyze the model's performance further. The success accuracy was calculated to be 97.91%, which is the percentage of instances the model correctly classified, and the failure accuracy was estimated to be 2.09%, which is the percentage of instances that the model incorrectly classified. Figure 5.1 below shows the results of this algorithm.



```
RF Classifier Model Train Score is :  0.9790822146091105
RF Classifier Model Test Score is :  0.979092221850675
confusion matrix
[[212681  13915]
 [    28 440257]]
succes accuracy = 97.91 %
fail accuracy = 2.09 %
------------------------------------------------------------------------------
```

Figure 5.1. The random forest classifier model results.

Overall, the high and low fail accuracy scores suggest that the model performs well on the dataset. However, further analysis may be required to understand why the model makes certain incorrect predictions and how the model's performance can be further improved.

5.2.2. K Nearest Neighbor

The following Figure 5.2 shows the results of the KNeighbors Classifier model.

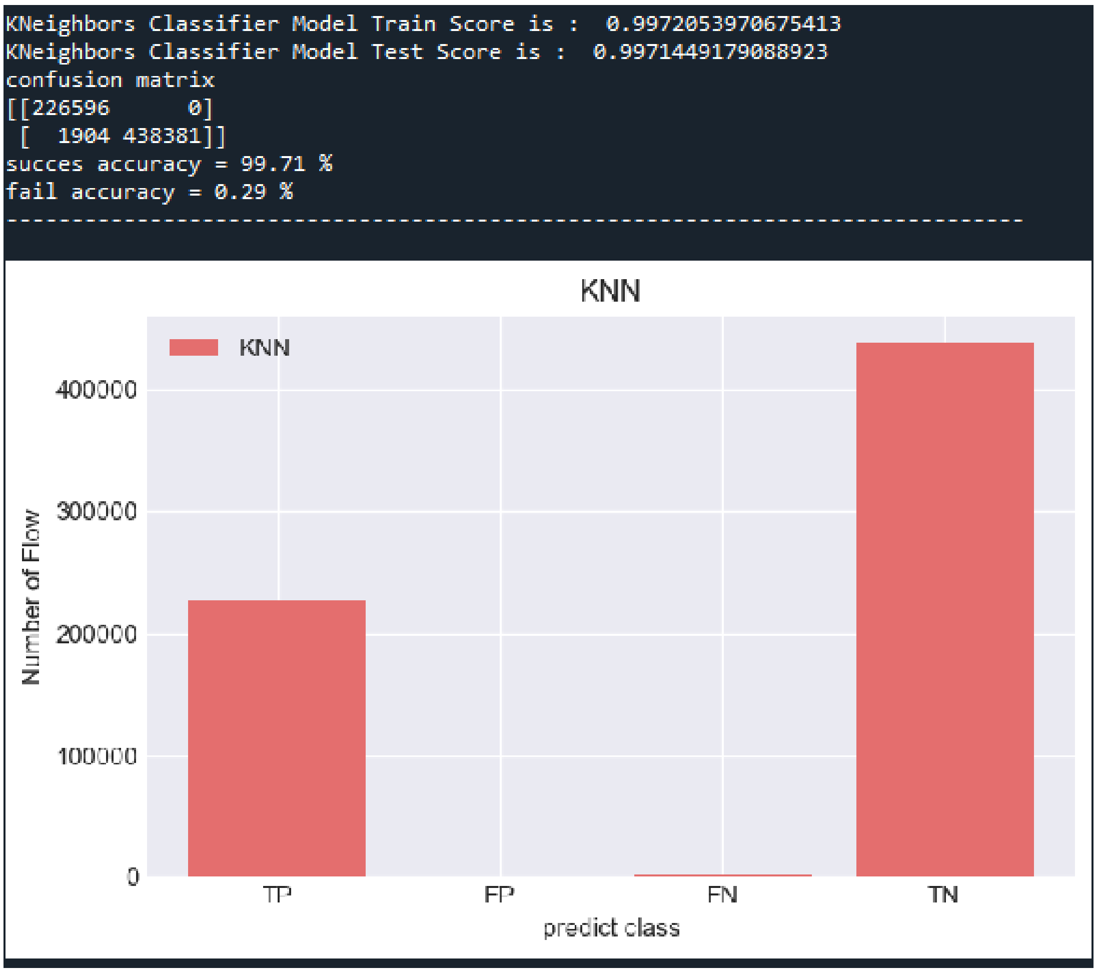

```
KNeighbors Classifier Model Train Score is :  0.9972053970675413
KNeighbors Classifier Model Test Score is :  0.9971449179088923
confusion matrix
[[226596      0]
 [  1904 438381]]
succes accuracy = 99.71 %
fail accuracy = 0.29 %
-----------------------------------------------------------------------------
```

Figure 5.2. The Kneighbors classifier model results.

The KNeighbors Classifier model was trained and evaluated using the proposed dataset. The training and test scores of the model were calculated as 0.9972 and 0.9971, respectively, indicating that the model performed well in both the training and testing phases.

The confusion matrix was used to assess the performance of the model. The matrix showed that out of 666,881 instances, 226,596 instances were predicted correctly as “success” and 438,381 instances were predicted correctly as “fail.” However, the model made 1,904 false predictions by classifying “fail” instances as “success.”

The success accuracy of the model was found to be 99.71%, indicating that the model correctly predicted “success” instances in the dataset with high accuracy. However, the failure accuracy of the model was only 0.29%, indicating that the model struggled to predict “fail” instances in the dataset.

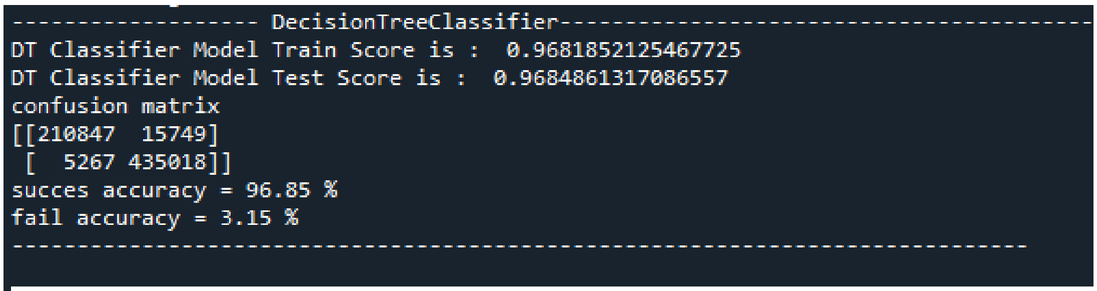
5.2.3. Decision Tree

The results presented in Figure 5.3 relate to the performance of a Decision Tree Classifier on a binary classification problem. The dataset was partitioned into training and testing sets, and the model was trained on the training data and evaluated on the testing data[77].

The Decision Tree Classifier achieved a training score of 0.9682, meaning it correctly classified 96.82% of the samples in the training data. The model also achieved a test score of 0.9685, correctly classifying 96.85% of the samples in the testing data. These scores suggest that the model can effectively generalize new data.

The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives for the model's predictions. The model correctly predicted that 435,018 samples belonged to the positive class (true positives), while incorrectly predicting that 5,267 samples belonged to the negative class (false negatives) and 15,749 samples belonged to the positive class when they belonged to the negative class (false positives). The model correctly predicted that 210,847 samples belonged to the negative (true negatives) class.

The success accuracy of the model, also known as the overall accuracy, is calculated as the number of correct predictions (true positives + true negatives) divided by the total number of predictions. In this case, the success accuracy is 96.85%. The failure accuracy of the model, also known as the error rate, is calculated as the number of incorrect predictions (false positives + false negatives) divided by the total number of predictions. In this case, the failure accuracy is 3.15%.



```
-------------------- DecisionTreeClassifier----------------------------------------
DT Classifier Model Train Score is :  0.9681852125467725
DT Classifier Model Test Score is :  0.9684861317086557
confusion matrix
[[210847  15749]
 [  5267 435018]]
succes accuracy = 96.85 %
fail accuracy = 3.15 %
------------------------------------------------------------------------------
```

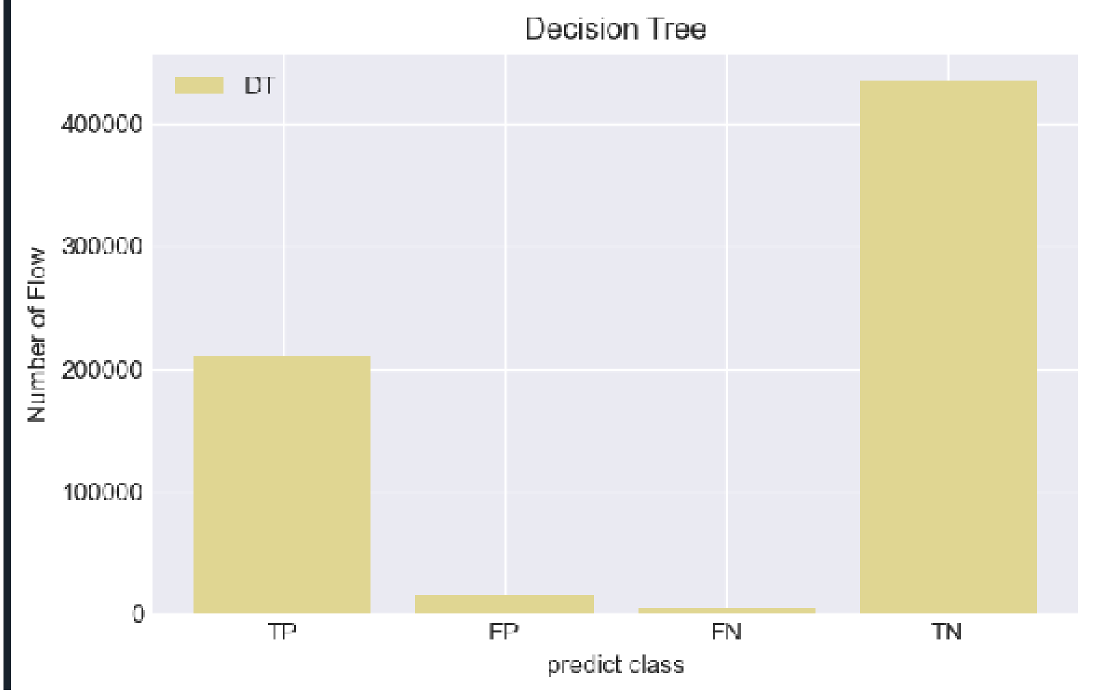
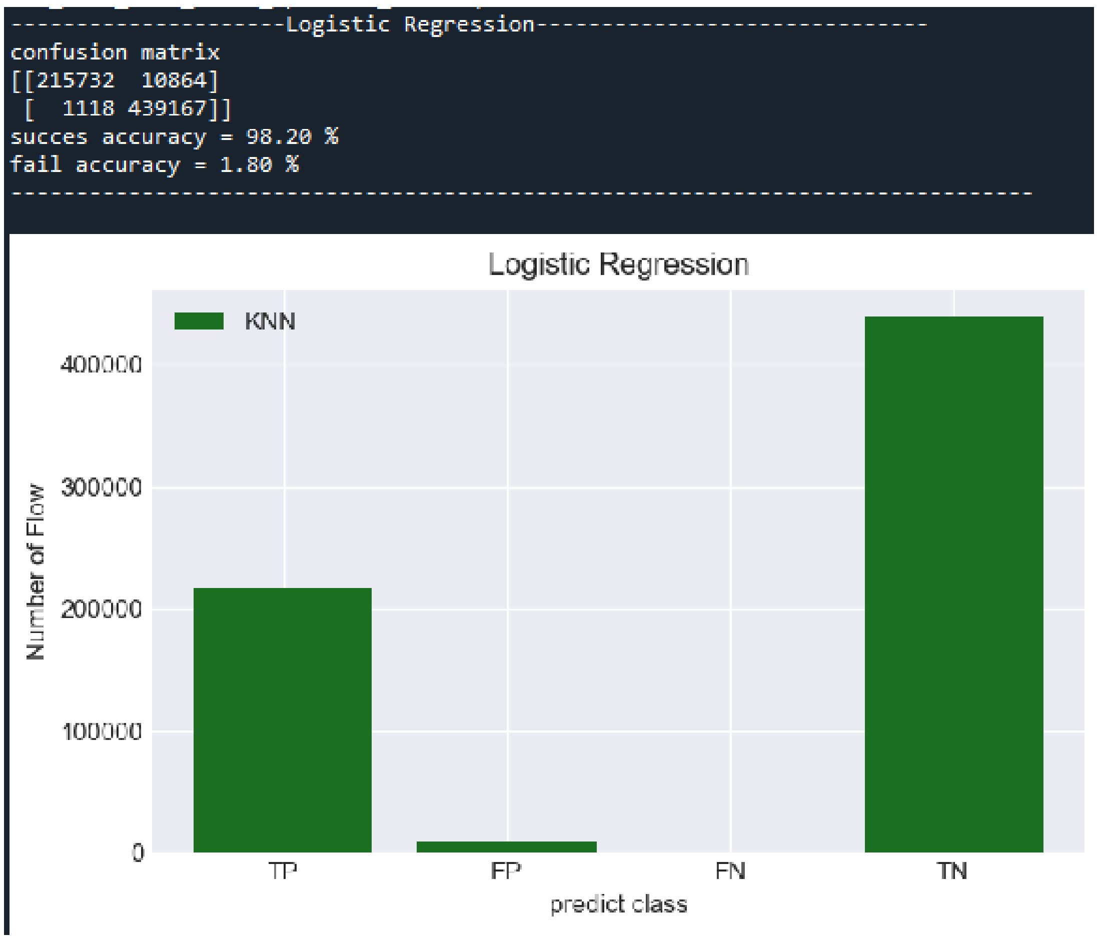


Figure 5.3. The decision tree classifier model results.

Overall, the results suggest that the Decision Tree Classifier can classify the binary target variable in the dataset with a high success accuracy and low error rate.

5.2.4. Logistic Regression

Figure 5.4 below shows the confusion matrix and accuracy scores for a logistic regression model on a binary classification task.



```
---------------------Logistic Regression------------------------------
confusion matrix
[[215732  10864]
 [  1118 439167]]
succes accuracy = 98.20 %
fail accuracy = 1.80 %
------------------------------------------------------------------------------
```

Figure 5.4. The logistic regression classifier model results.

In this case, the model classified 215,732 samples as successful (true negatives) and 439,167 as unsuccessful (true positives). However, the model misclassified 1,118 unsuccessful samples as successful (false positives) and 10,864 successful samples as unsuccessful (false negatives).

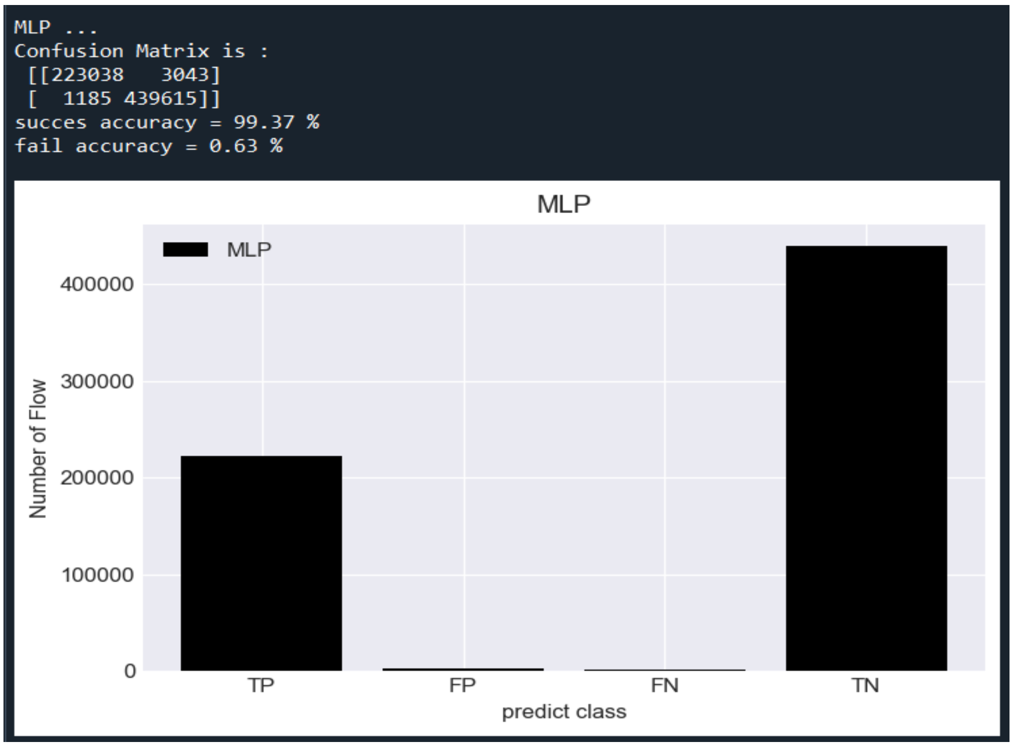
The "success accuracy" of 98.20% indicates that the model correctly classified 98.20% of the successful samples in the testing dataset. In comparison, the "fail accuracy" of 1.80% suggests the model correctly classified only 1.80% of the unsuccessful samples in the testing dataset.

These accuracy scores provide a more detailed evaluation of the model's performance for each class in the classification problem. In general, accuracy scores are used to measure the effectiveness of a classification model in terms of its ability to predict class labels accurately. High accuracy scores indicate that the model makes correct predictions for the given classification issue.

5.2.5. Multi-Layer Perceptron (MLP)

The MLP Classifier model was trained and evaluated on the dataset, and the confusion matrix in Figure 5.5 shows that the model is performing exceptionally well. It made 223,038 correct predictions of true negatives and 439,615 correct predictions of true positives but also made 3,043 incorrect predictions of false positives and 1,185 incorrect predictions of false negatives.

The success accuracy of the model is 99.37%, indicating that the model correctly predicted the class labels for 99.37% of all instances in the dataset. The failure accuracy is 0.63%, meaning that the model misclassified 0.63% of instances in the dataset.



```
MLP ...
Confusion Matrix is :
 [[223038   3043]
 [  1185 439615]]
succes accuracy = 99.37 %
fail accuracy = 0.63 %
```

Figure 5.5. The MLP classifier model result.

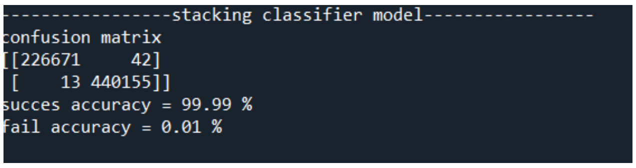
5.2.6. Stacking Classifier

Figure 5.6 presents the findings of an analysis that examined how well a stacking classifier model performed on a binary classification dataset. The performance of a classification model may be evaluated with the use of a table called the confusion matrix. This matrix works by contrasting the predicted class labels with the actual class labels.

In this particular scenario, the confusion matrix reveals that the stacking classifier model has generated 226,671 true positive predictions (that is, the model accurately predicted the positive class) and 440,155 true negative predictions (i.e., the model correctly predicted the negative class). There are also 42 erroneous negative predictions, which means that the model mistakenly predicted the negative class while the actual class was positive. On the flip side, there are 13 inaccurate positive predictions (i.e., incorrectly predicted the positive class when the actual class was negative).

To get the success accuracy of a model, just take the ratio of accurate forecasts to the total number of predictions produced by the model, then multiply that number by 100. In this instance, the success accuracy is 99.99 percent, which suggests that the stacking classifier model works very well in properly predicting the class labels. This is shown by the fact that the percentage of successful predictions is so high.

A stacking classifier model is one that combines the predictions of numerous base classifiers into one. This helps to increase the model's overall accuracy as well as its resilience. The findings indicate that the basic classifiers used in the stacking classifier model are operating well, and that their predictions are being merged in an efficient manner to provide correct outcomes.



```
----------------stacking classifier model----------------
confusion matrix
[[226671     42]
 [    13 440155]]
succes accuracy = 99.99 %
fail accuracy = 0.01 %
```

Figure 5.6. The stacking classifier model result.

5.3. THE VALIDATION AND EVALUATION OF MACHINE LEARNING MODELS

k-fold cross-validation is an approach that is often used in the process of assessing machine learning models. This method requires splitting the dataset into k subsets of equal size, training the model on k-1 subsets, and utilizing the remaining subset for testing, as is seen in Figure 5.7. This method is carried out k times, with each iteration employing a unique subset for testing. The findings of these iterations are then averaged together to get the final assessment metrics. In order to evaluate the effectiveness of the models, it is common practice to utilize evaluation measures such as accuracy, sensitivity, specificity, and precision, as well as the F1 score [84], [90], [95].

Each fold must have these features in order to guarantee that the majority of the categorization attributes are used in both the training and the testing phases [96]. As the ultimate predictive and definitive validation strategy, the 5-fold cross-validation technique was used. An independent test set is carried out using this strategy [96], [97].

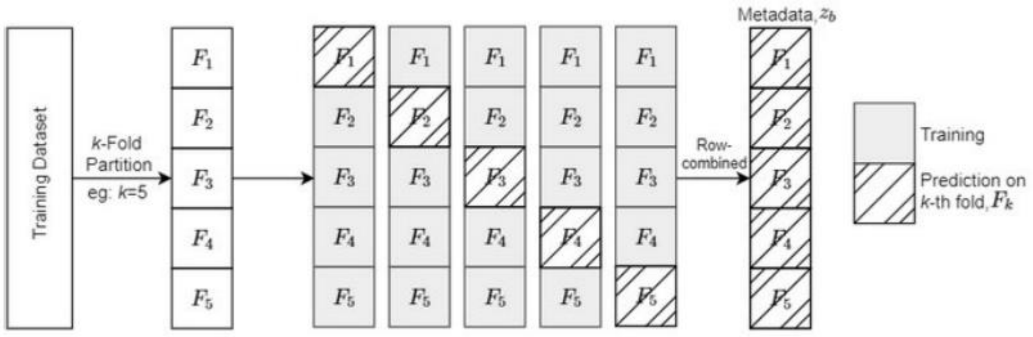


Figure 5.7. k-fold cross-validation for the generation of metadata, zb , k = 5 [98].

In this study, the performance of five popular machine learning algorithms was evaluated using 5-fold cross-validation. These algorithms were Random Forest (RF), Decision Tree (DT), Logistic Regression (LR), Multi-Layer Perceptron (MLP), and K-Nearest Neighbors (KNN). In addition, a Stacking Classifier was used to combine the predictions of these models.

Accuracy, sensitivity, specificity, and precision were the evaluation metrics that were taken into consideration for this research. The F1 score was also taken into account. The use of k-fold cross-validation in conjunction with these evaluation measures was deemed to be appropriate due to the general acceptance of both, as well as their capacity to provide a reliable and all-encompassing evaluation of model performance. It is important to keep in mind, however, that in k-fold cross-validation, increasing or lowering the number of folds does not always ensure a more accurate result every time. There is a complicated link between the number of folds and the accuracy of the model; increasing the number of folds may result in an increase in computing expenses without necessarily leading to an increase in accuracy [99].

Table 5.1 Performance comparison of machine learning models for a binary classification task.

Model	Accuracy	Sensitivity	Specificity	Precision	F1 Score
Random forest	97.91%	99.98%	93.88%	96.95%	98.44%
K nearest neighbor	99.73%	99.58%	100.00%	100.00%	99.79%
Decision tree	96.85%	98.80%	93.05%	96.51%	97.64%
Logistic regression	98.20%	99.75%	95.18%	97.57%	98.65%
MLP	98.77%	98.40%	99.48%	99.73%	99.06%
Stacking model	99.99%	100.00%	99.98%	99.99%	99.99%

As can be seen in Table 5.1, six distinct machine learning models were used to make predictions on the proposed dataset. The performance of each model was evaluated using a variety of metrics on a separate test set that was not used during the training process. This test set was kept separate from the training set. The use of this method helps to guarantee that the outcomes of the assessment are trustworthy and impartial.
The stacking model achieved the maximum level of accuracy, 99.99 %, based on the findings that were provided. This suggests that the stacking model has the potential to appropriately categorize DDoS assaults in an SDN setting. Additionally, the MLP model did quite well, with an accuracy of 98.77 %, which is encouraging. The decision tree model, on the other hand, had the lowest accuracy of all the models, which implies

that it is not necessarily the most suitable option for this particular application. Accuracy alone is not always a good metric for evaluating the performance of a model, and it is important to note that other metrics such as sensitivity, specificity, precision, and F1 score should also be considered. It is important to note that accuracy alone is not always a good metric for evaluating the performance of a model.

The next metric reported is the sensitivity of each model, which is the ability of the model to identify true positive cases. A higher sensitivity indicates that the model is better at correctly identifying positive cases, vital in applications where false negatives can have serious consequences.

Based on the results, the stacking model has the highest sensitivity of 100%, correctly identifying all true positive cases. This is followed closely by the random forest model with a sensitivity of 99.98% and the logistic regression model with a sensitivity of 99.75%. The K nearest neighbor and MLP models have lower sensitivity scores, indicating they may be less effective in correctly identifying positive cases. The decision tree model has the lowest sensitivity of 98.80%.

Based on the specific results, K nearest neighbor, logistic regression, MLP, and stacking model seem to have high values, indicating that they can effectively identify true negative cases of no DDoS attack in SDN. On the other hand, random forest and decision tree models have relatively lower values, indicating that they may have a higher chance of falsely identifying regular traffic as a DDoS attack.

Precision measures the proportion of true positive predictions out of all positive predictions made by the model. In detecting DDoS attacks in SDN, precision is a critical metric because it measures the model's accuracy in identifying real attacks while avoiding false positives.

The results show that all models achieved high precision values ranging from 96.95% to 100%. This suggests that all models effectively minimize the number of false positives, which is crucial for reducing the risk of disrupting legitimate network traffic. Moreover, the K nearest neighbor model achieved the highest precision value of 100%.

This means that all positive predictions made by this model were true positives, indicating that it is the most accurate model in detecting DDoS attacks in SDN.

In general, the precision values obtained by all models indicate that they are well-suited for detecting DDoS attacks in SDN and can be used in practice to enhance network security. The F1 score is a measure of a model's accuracy that takes both precision and recall into account. The higher the F1 score, the better the model's overall performance. Looking at the F1 scores of the models, the K nearest neighbor model has the highest score of 99.79%. This suggests it has the best overall balance between precision and recall for detecting DDOS attacks in SDN. The stacking model also has a high F1 score of 99.99%, indicating that it performs very well in overall accuracy. The MLP and logistic regression models also have high F1 scores of 99.06% and 98.65%, respectively, indicating that they are also effective in detecting DDOS attacks in SDN. The decision tree and random forest models have lower F1 scores of 97.64% and 98.44%, respectively, suggesting they may not be as effective as the other models in detecting DDOS attacks in SDN. These results indicate that the stacking classifier is a highly effective ML algorithm for the problem. Its superior performance can be attributed to combining the predictions of multiple models, each of which may have different strengths and weaknesses. By leveraging the strengths of other models, the stacking classifier can achieve higher accuracy and robustness compared to any single model alone. The resulting bar chart shows the values of different evaluation metrics for various machine learning models. The x-axis represents the names of the models, while the y-axis shows the importance of metrics such as accuracy, sensitivity, specificity, precision, and F1 score.

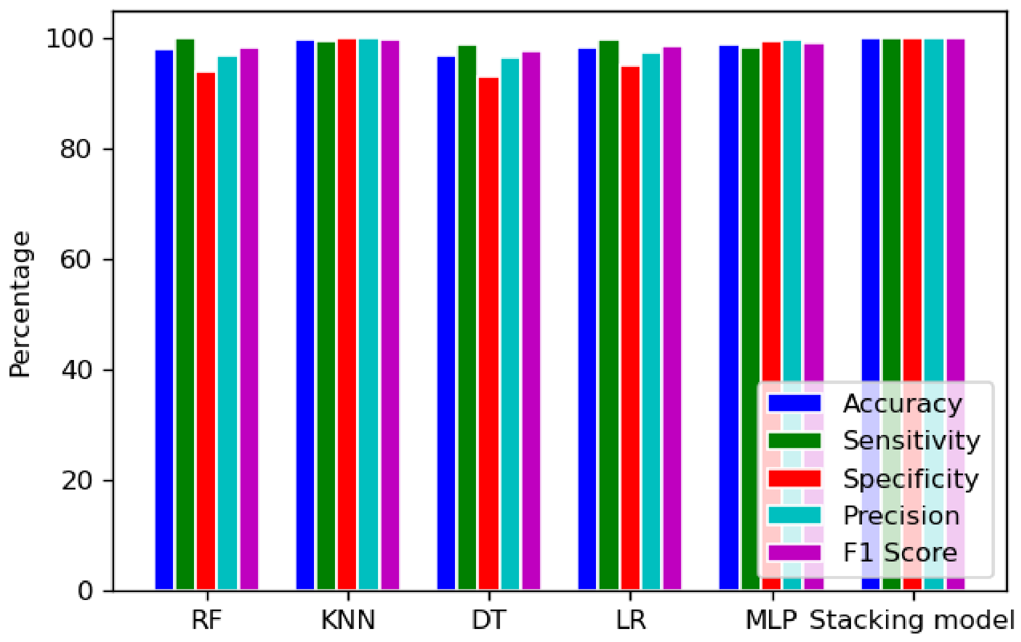


Figure 5.8. Comparison of ml models based on multiple evaluation metrics.

Based on the findings presented in Figure 5.8, it is evident that the stacking model exhibits superior performance over all other models evaluated, as indicated by its higher values for all the considered evaluation metrics. Furthermore, the K Nearest Neighbor model demonstrates good performance, having achieved perfect scores for sensitivity, specificity, and F1, albeit with marginally lower accuracy than the stacking model.

The Random Forest model has a high accuracy but a lower sensitivity and precision than the Stacking and K Nearest Neighbor models. The Logistic Regression and MLP models also have high values for all metrics but are slightly lower than those of the Stacking and K Nearest Neighbor models.

Finally, the Decision Tree model has the lowest accuracy and sensitivity among all the models, although its specificity, precision, and F1 score are still relatively high. Overall, the graph provides a quick and easy way to compare the performance of different machine learning models on multiple evaluation metrics and can help choose the best model for a given task.

5.4. RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

The Receiver Operating Characteristic Curve, often known as the ROC Curve, is a graphical figure that depicts the performance of a binary classification model [95]. It generates a graphic that compares the true positive rate (TPR) to the false positive rate (FPR) using a variety of threshold values. The true positive rate, or sensitivity, is the proportion of genuine positive samples that the model properly identifies, while the false positive rate, or FPR, is the proportion of actual negative samples that are wrongly categorized as positive. Both measures are expressed as a percentage. The performance of binary classifiers is often evaluated using the Area Under the ROC Curve (AUC), which is an alternative name for the ROC area. The area under the receiver operating characteristic curve for a perfect classifier is 1, whereas the AUC for a random classifier is 0.5 [86].

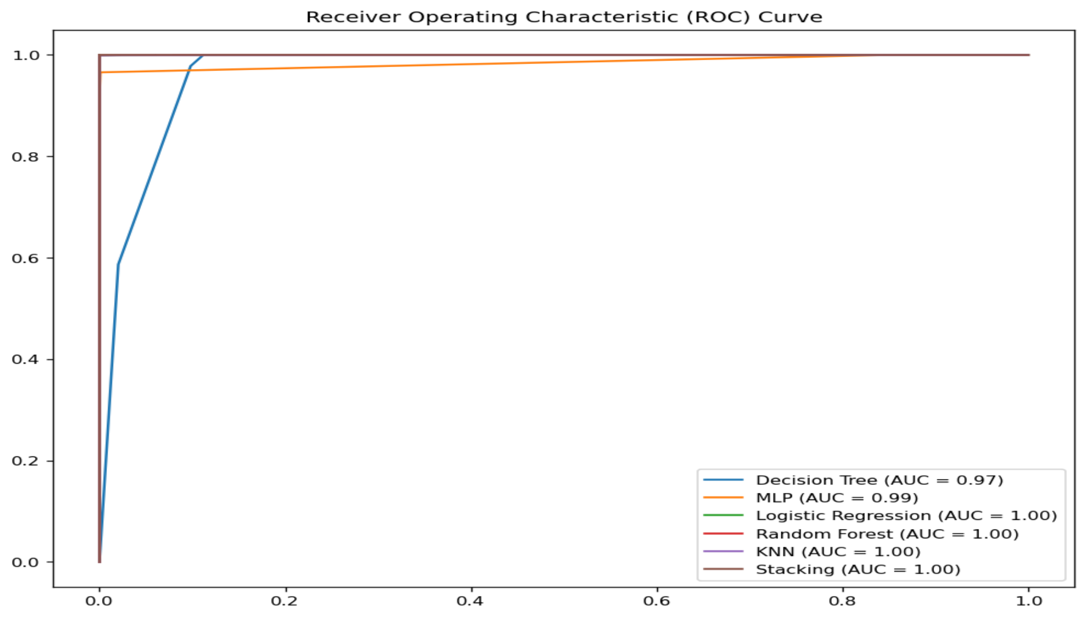


Figure 5.9. The receiver operating characteristic (ROC) curve for various classification models.

The ROC curve for the training and testing sets was produced by plotting the TPR against the FPR at various classification thresholds, as can be seen in the Figure 5.9 that was just shown. The ROC curve is constructed within the context of the prior code in order to assess the performance of the recommended DDoS attack detection model. The ROC curve illustrates the degree to which the model is able to differentiate

between a DDoS attack and typical network traffic. The performance of the model is said to be superior the closer the ROC curve is to the top left corner of the graph.
In addition, it would seem that all of the models, including MLP, DT, RF, KNN, LR, and Stacking, perform remarkably well in identifying DDoS assaults in an SDN environment, with AUC values ranging from 0.97 to 1.00. This conclusion is based on the AUC values that are shown in the graph. A model is considered to have flawless classification performance if it has an AUC value of 1.0, which implies that it is able to differentiate between normal and malicious traffic in an accurate manner.

It is important to point out that the stacking model has an AUC value of 1.00, which indicates that it is doing even better than the separate models. Stacking is a method that combines the predictions that are produced by numerous models in order to get a final forecast that is more accurate. The fact that the stacking model is doing so well provides strong evidence that the separate models are additive to one another and may be merged to get a more accurate forecast of the whole.

Overall, the ROC curve is a powerful tool for evaluating the performance of binary classification models [99], including DDoS attack detection methods. It provides a visual representation of the tradeoff between detection accuracy and false alarms and allows us to choose the threshold that best balances these two factors.

5.5. CONCLUDING REMARKS

In summary, this chapter has presented the results of evaluating various machine learning classifier models for DDoS detection. The Confusion Matrix provided a detailed analysis of the system's performance, while the Classifier Models Results demonstrated the success accuracy of each model, ranging from 96.85% to 99.99%. The Validation and Evaluation of Machine Learning Models offered insights into the performance of these models. Finally, the Receiver Operating Characteristic (ROC) Curve showed the superiority of the Stacking Classifier model, which outperformed the individual models with an AUC value of 1.00. These findings are significant for developing effective DDoS detection systems that can enhance network security.

PART 6

CONCLUSION AND FUTURE WORK

This chapter provides a summary of the key findings and contributions made to the field of network security, specifically in the context of software-defined networking (SDN) and DDoS attack detection. Additionally, highlights potential areas for future research and development in the field, to further enhance the security and resilience of SDN networks against DDoS attacks.

6.1. RESEARCH SUMMARY

In contemporary times, networks have emerged as an indispensable element in facilitating various activities. They provide vital communication links that enable organizations to operate their applications and maintain competitiveness. With the increasing demand for high data rates and real-time applications, the networking industry is confronted with the challenge of constantly reconfiguring already complex, vendor-specific equipment, which offers little or no flexibility and interoperability.

SDN is an open technology that holds promise for more innovative, flexible, and effective solutions. While SDN presents a simple framework for network programmability and monitoring, its security remains a significant aspect, and the potential impact of a security breach cannot be overemphasized. Despite the rapid pace of research into network security in the scientific and mathematical world, its applications and practicality have not kept pace.

The research aims to address the issue of DDoS attacks in SDN by proposing a comprehensive approach for detecting network attacks. The research reviews the vulnerabilities of SDN to DDoS attacks and the efforts made to detect and mitigate them. To achieve this goal, a novel network topology was designed and implemented, which enabled the extraction of real-time SDN traffic data before and after a DDoS attack to generate a unique dataset. The proposed solution involves the use of three main modules: Feature Creation, Stacking Classifier, and Attack Detection, along with various environmental requirements. The classification methodology includes multiple ML/DL algorithms, and the implementation of the Grid search hyperparameter tuning method improves the overall performance of the solution. The results demonstrate the success accuracy of each model, ranging from 96.85% to 99.99%, and the superiority of the Stacking Classifier model for developing effective DDoS detection systems that can enhance network security.

However, a DDoS attack with a higher number of active agents can result in greater severity. Therefore, the implementation of a sturdy and resilient security architecture for SDN is imperative. Although assessing the impact of DDoS attacks on SDNs is a challenging task, the present research provides a preliminary framework for conducting an objective evaluation of DDoS attacks on SDNs.

6.2. RESEARCH CONTRIBUTIONS

This research contributes to the field of network security, specifically in the context. of SDN.

1. Identify and analyze the different types of DDoS attacks that can affect SDN networks, as well as evaluate the existing SDN security solutions that can be used to mitigate them. This can lead to a better understanding of the security challenges faced by SDN networks and the effectiveness of current solutions, which can inform the development of new and more effective security measures.
2. An end-to-end intrusion detection system architecture is proposed for SDN-based networks. This architecture tracks traffic flows at the controller, classifies traffic using a machine learning-based intelligent attack detection module and modifies flow rules accordingly.
3. a novel SDN dataset is generated that includes regular traffic and various attack traffic types.

4. The Stacking Classifier technique is introduced for detecting DDoS attacks in SDN, which combines the predictions of multiple base classifiers to improve classification performance. To the best of our knowledge, this is the first study that applies the stacking classifier technique to detect DDoS attack in SDN. These findings suggest that incorporating the stacked classifier technique may enhance the accuracy and robustness of DDoS attack detection models in SDN, making it a valuable contribution to the field of network security.
5. The approach offers significant advantages over existing solutions, including direct protection for the controller against DDoS attacks, a focus specifically on SDN environments, and a high accuracy rate of 99%. Finally, the detection algorithm is successfully implemented in both the RYU controller and Mininet, demonstrating the practical utility of the proposed approach.

The diagram presented in Figure 6.1 illustrates the interrelation between the contributions of this study, the corresponding research questions that have been addressed, and the way they have been addressed in the ensuing chapters.

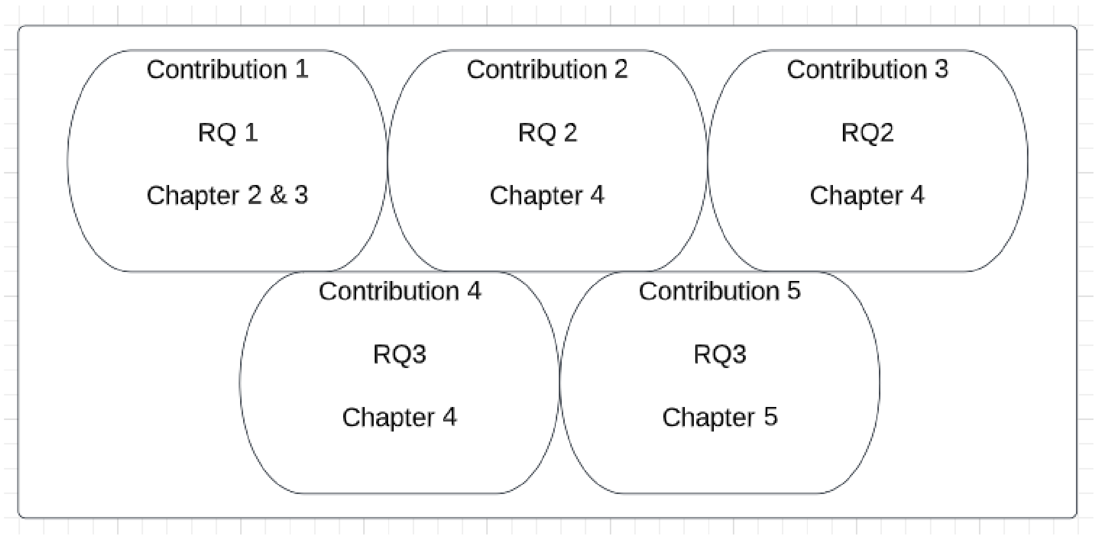


Figure 6.1. Interrelation between research questions and chapters.

6.3. RESEARCH LIMITATIONS

1. The research only focused on the binary classification of attacks. A multi-class classification approach can be used to detect and differentiate between different types of cyberattacks.

2. The research only considered DDoS attacks targeting individual hosts. DDoS attacks targeting network subnets can have larger impacts, but also pose additional challenges for detection and mitigation.

3. The research did not assess the impacts of DDoS attacks launched through external compromised networks (botnets). Such DDoS attacks can potentially cause damage on a much larger scale.

6.4. FUTURE WORK

The thesis presented herein proposes several areas for future research and expansion within the field of software-defined networking (SDN) security. This section outlines specific recommendations for future work, including proposals to expand upon the current study and suggestions for entirely new areas of inquiry. These recommendations serve to guide and inform future research efforts, with the goal of advancing our understanding of the challenges and potential solutions related to SDN security.

6.4.1. Self-healing

A timely error log helps reduce network downtime due to the SDN controller's global view advantage**.** Network self-healing is attractive because service outages and other network-related problems may be fixed immediately without the need for network administrators to become involved [100]. Fewer network managers will be needed in self-healing networks provided by SDN, saving the cost for organizations and higher customer satisfaction. Several studies on self-healing networks regarding SDN may be found in [101][102][103] [104].

6.4.2. Load Balancing

Resource allocation is addressed via load balancing, which ensures that the use of available resources is effective [105]. During a DDoS attack, an overloaded link can be avoided by locating underutilized paths and distributing traffic among them [106]. The following are some studies on load balancing [107] [108].

6.4.3. Integration to Cloud

Network resources are now available on demand due to cloud computing technology, which provides pay-as-you-go services [109] [110]. SDN-enabled cloud services will provide elastic services to clients with high availability, and the SDN controller can efficiently manage cloud networking resources [39] [111].

6.4.4. Database Optimization

We believe the dataset generated in this study will be a valuable resource for future research on ML-based intrusion detection in SDN networks. A potential avenue for future work involves expanding the generated dataset to include additional attack types and network topologies.

6.4.5. Exploring the Impacts of DDoS Attacks Launched Through External Networks

In the future, it would be valuable to study the impacts of DDoS attacks that come from outside a targeted system or network. Currently, research on DDoS attacks has mostly focused on those that originate from within the targeted system. However, attacks from external networks, such as botnets or other compromised systems, are becoming increasingly common and can be just as damaging. Therefore, studying these types of attacks could provide important insights into the threat landscape and inform strategies for defending against them.

6.4.6. Enhancing Performance of Security Models Through Human-in-the-Loop (HITL) and Continuous Learning

Even though the suggested method has achieved good results in the network environment it was trained for, adaptability to other networks would necessitate actively training the model through online learning**.** However, through continuous learning, this will enable not just the ability to recognize attack types that have already been identified, but also the capability to classify recent attacks correctly.
On the other hand, while certain transfer learning techniques are effective, in many situations they fall short of the performance of ML models trained on data collected from real operational environments. So, future work will focus on building continuous learning with a Human-In-The-Loop (HITL) system [112] to achieve high performance in various network structures.

6.4.7. Secure Network Control

Blockchain can create a decentralized and tamper-resistant network control plane [113] . Instead of relying on a central controller, blockchain-based SDN can distribute the control plane across multiple nodes, making it more resilient against single points of failure or attacks on the controller. Consensus algorithms, such as Proof of Work (PoW) or Proof of Stake (PoS), can validate and agree upon network control decisions [114].

6.4.8. Identity and Access Management

Blockchain can enhance identity and access management (IAM) in an SDN environment [115]. By using blockchain-based identity systems, such as self-sovereign identity (SSI) or decentralized identifiers (DIDs), it becomes possible to establish secure and verifiable identities for network devices, users, and applications [116]. This can strengthen authentication and authorization mechanisms in SDN, ensuring that only trusted entities have access to the network.

REFERENCES

[1] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] M. W. Nadeem, H. G. Goh, V. Ponnusamy, and Y. Aun, "Ddos detection in sdn usingmachine learning techniques," *Computers, Materials and Continua*, vol. 71, no. 1, pp. 771–789, 2022, doi: 10.32604/cmc.2022.021669.

[3] A. Sangodoyin, T. Sigwele, P. Pillai, Y. F. Hu, I. Awan, and J. Disso, "DoS attack impact assessment on software defined networks," in *Wireless and Satellite Systems: 9th International Conference, WiSATS 2017, Oxford, UK, September 14-15, 2017, Proceedings 9*, Springer, 2018, pp. 11–22.

[4] S. Das, G. Parulkar, and N. McKeown, "Rethinking IP core networks," *Journal of Optical Communications and Networking*, vol. 5, no. 12, pp. 1431–1442, 2013.

[5] E. Molina and E. Jacob, "Software-defined networking in cyber-physical systems: A survey," *Computers & electrical engineering*, vol. 66, pp. 407–419, 2018.

[6] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, pp. 101–118, 2018.

[7] A. Mondal, S. Misra, and I. Maity, "AMOPE: Performance analysis of OpenFlow systems in software-defined networks," *IEEE Syst J*, vol. 14, no. 1, pp. 124–131, 2019.

[8] M. Conti, C. Lal, R. Mohammadi, and U. Rawat, "Lightweight solutions to counter DDoS attacks in software defined networking," *Wireless Networks*, vol. 25, pp. 2751–2768, 2019.

[9] C. B. Zerbini, L. F. Carvalho, T. Abrão, and M. L. Proenca Jr, "Wavelet against random forest for anomaly mitigation in software-defined networking," *Appl Soft Comput*, vol. 80, pp. 138–153, 2019.

[10] O. Dakkak, S. A. Nor, and S. Arif, “Scheduling Jobs through Gap Filling and Optimization Techniques in Computational Grid.,” *J. Comput. Sci.*, vol. 13, no. 5, pp. 105–113, 2017.

[11] O. Dakkak, S. A. Nor, and S. Arif, “Proposed algorithm for scheduling in computational grid using backfilling and optimization techniques,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 8, no. 10, pp. 133–138, 2016.

[12] S. M. Mousavi and M. St-Hilaire, “Early Detection of DDoS Attacks Against Software Defined Network Controllers,” *Journal of Network and Systems Management*, vol. 26, no. 3, pp. 573–591, Jul. 2018, doi: 10.1007/s10922-017-9432-1.

[13] Y. Xu and Y. Liu, “DDoS attack detection under SDN context,” in *IEEE INFOCOM 2016-the 35th annual IEEE international conference on computer communications*, IEEE, 2016, pp. 1–9.

[14] A. Akhunzada, E. Ahmed, A. Gani, M. K. Khan, M. Imran, and S. Guizani, “Securing software defined networks: taxonomy, requirements, and open issues,” *IEEE Communications Magazine*, vol. 53, no. 4, pp. 36–44, 2015.

[15] N. McKeown *et al.*, “OpenFlow: enabling innovation in campus networks,” *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[16] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: Taking control of the enterprise,” *ACM SIGCOMM computer communication review*, vol. 37, no. 4, pp. 1–12, 2007.

[17] H. Al Mansoori, A. Bin Ali, and M. K. al Hassan, “Nature and Quality of Smart Government Services: The Case of the UAE,” *Int J Enhanc Res Sci Technol Eng*, vol. 5, no. 1, pp. 53–64, 2016.

[18] S. Jain *et al.*, “B4: Experience with a globally-deployed software defined WAN,” *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[19] “Network Virtualization and Security Software.”.

[20] “THE LINUX FOUNDATION PROJECTS”.

[21] S. Sezer *et al.*, “Are we ready for SDN? Implementation challenges for software-defined networks,” *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[22] D. Kreutz, F. M. V Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-defined networking: A comprehensive survey,” *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[23] O. Dakkak, S. A. Nor, and S. Arif, “Scheduling through backfilling technique for HPC applications in grid computing environment,” in *2016 IEEE Conference on Open Systems (ICOS)*, IEEE, 2016, pp. 30–35.

[24] J. C. Correa Chica, J. C. Imbachi, and J. F. Botero Vega, “Security in SDN: A comprehensive survey,” *Journal of Network and Computer Applications*, vol. 159. Academic Press, Jun. 01, 2020. doi: 10.1016/j.jnca.2020.102595.

[25] T. ALASALI and O. DAKKAK, “EXPLORING THE LANDSCAPE OF SDN-BASED DDOS DEFENSE: A HOLISTIC EXAMINATION OF DETECTION AND MITIGATION APPROACHES, RESEARCH GAPS AND PROMISING AVENUES FOR FUTURE EXPLORATION,” *International Journal of Advanced Natural Sciences and Engineering Researches*, vol. 7, no. 4, pp. 327–349, 2023.

[26] T. Han *et al.*, “A comprehensive survey of security threats and their mitigation techniques for next-generation SDN controllers,” in *Concurrency and Computation: Practice and Experience*, John Wiley and Sons Ltd, Aug. 2020. doi: 10.1002/cpe.5300.

[27] B. Pavlyshenko, “Using stacking approaches for machine learning models,” in *2018 IEEE second international conference on data stream mining & processing (DSMP)*, IEEE, 2018, pp. 255–258.

[28] Open Networking Foundation, “Open Networking Foundation. Software-Defined Networking: The New Norm for Network,” *https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/.*

[29] Y. Y. Baydilli and İ. Türker, “Is the world small enough?—A view from currencies,” *Int J Mod Phys B*, vol. 33, no. 12, p. 1950120, 2019.

[30] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “A Survey on Software-Defined Networking,” *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1. Institute of Electrical and Electronics Engineers Inc., pp. 27–51, Jan. 01, 2015. doi: 10.1109/COMST.2014.2330903.

[31] H. Polat, O. Polat, and A. Cetin, “Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models,” *Sustainability (Switzerland)*, vol. 12, no. 3, Feb. 2020, doi: 10.3390/su12031035.

[32] K. M. Sudar, M. Beulah, P. Deepalakshmi, P. Nagaraj, and P. Chinnasamy, “Detection of Distributed Denial of Service Attacks in SDN using Machine learning techniques,” in *2021 International Conference on Computer Communication and Informatics, ICCCI 2021*, Institute of Electrical and Electronics Engineers Inc., Jan. 2021. doi: 10.1109/ICCCI50826.2021.9402517.

[33] M. Ahmed, S. Shatabda, A. K. M. Islam, M. Robin, and T. Islam, “Intrusion Detection System in Software-Defined Networks Using Machine Learning and Deep Learning Techniques–A Comprehensive Survey,” 2021.

[34] B. Rauf *et al.*, “Application Threats to Exploit Northbound Interface Vulnerabilities in Software Defined Networks,” *ACM Computing Surveys*, vol. 54, no. 6. Association for Computing Machinery, Jul. 01, 2021. doi: 10.1145/3453648.

[35] W. Braun and M. Menth, “Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices,” *Future Internet*, vol. 6, no. 2, pp. 302–336, May 2014, doi: 10.3390/fi6020302.

[36] C. Fan, N. M. Kaliyamurthy, S. Chen, H. Jiang, Y. Zhou, and C. Campbell, “Detection of DDoS Attacks in Software Defined Networking Using Entropy,” *Applied Sciences (Switzerland)*, vol. 12, no. 1, Jan. 2022, doi: 10.3390/app12010370.

[37] D. Melkov and S. Paulikas, “Security Benefits and Drawbacks of Software-Defined Networking,” in *2021 IEEE Open Conference of Electrical, Electronic and Information Sciences, eStream 2021 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., Apr. 2021. doi: 10.1109/eStream53087.2021.9431466.

[38] M. M. Salim, S. Rathore, and J. H. Park, “Distributed denial of service attacks and its defenses in IoT: a survey,” *J Supercomput*, vol. 76, pp. 5320–5363, 2020.

[39] O. Dakkak, S. A. Nor, M. S. Sajat, Y. Fazea, and S. Arif, “From grids to clouds: Recap on challenges and solutions,” in *AIP Conference Proceedings*, AIP Publishing LLC, 2018, p. 020040.

[40] K. Srinivasan, A. Mubarakali, A. S. Alqahtani, and A. Dinesh Kumar, “A survey on the impact of DDoS attacks in cloud computing: prevention, detection and mitigation techniques,” in *Intelligent Communication Technologies and Virtual Mobile Networks: ICICV 2019*, Springer, 2020, pp. 252–270.

[41] A. Habbal, S. I. Goudar, and S. Hassan, “Context-aware radio access technology selection in 5G ultra dense networks,” *IEEE access*, vol. 5, pp. 6636–6648, 2017.

[42] D. Gurusamy, M. Deva Priya, B. Yibgeta, and A. Bekalu, “DDoS risk in 5G enabled IoT and solutions,” *Int J Eng Adv Technol*, vol. 8, no. 5, pp. 1574–1578, 2019.

[43] “ Kaspersky. (2021). Kaspersky Q4 2020 DDoS Attacks Report. [Online].”

[44] “2020-Q1_AWS_Shield_TLR”.

[45] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Perez-Diaz, “SDN-based architecture for transport and application layer DDoS attack detection by using

machine and deep learning," *IEEE Access*, vol. 9, pp. 108495–108512, 2021, doi: 10.1109/ACCESS.2021.3101650.

[46] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, Apr. 2004, doi: 10.1016/j.comnet.2003.10.003.

[47] L. Xinlong and C. Zhibin, "Ddos attack detection by hybrid deep learning methodologies," *Security and Communication Networks*, vol. 2022, 2022.

[48] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, Apr. 2004, doi: 10.1016/j.comnet.2003.10.003.

[49] M. D. Prasad, P. B. V, and C. Amarnath, "Machine Learning DDoS Detection Using Stochastic Gradient Boosting," *International Journal of Computer Sciences and Engineering*, vol. 7, no. 4, pp. 157–166, Apr. 2019, doi: 10.26438/ijcse/v7i4.157166.

[50] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research Trends in Security and DDoS in SDN," *Security and Communication Networks*, vol. 9, no. 18, pp. 6386–6411, Dec. 2016, doi: 10.1002/sec.1759.

[51] "OpenFlow Switch Specification Version 1.5.1 (Protocol version 0x06) for information on specification licensing through membership agreements," 2015. [Online]. Available: http://www.opennetworking.org

[52] Y.-L. Hu, W.-B. Su, L.-Y. Wu, Y. Huang, and S.-Y. Kuo, "Design of event-based Intrusion Detection System on OpenFlow Network," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2013, pp. 1–2. doi: 10.1109/DSN.2013.6575335.

[53] F. O. Catak and A. F. Mustacoglu, "Distributed denial of service attack detection using autoencoder and deep neural networks," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 3, pp. 3969–3979, 2019.

[54] Y. Wang, T. Hu, G. Tang, J. Xie, and J. Lu, "SGS: Safe-Guard Scheme for Protecting Control Plane Against DDoS Attacks in Software-Defined Networking," *IEEE Access*, vol. 7, pp. 34699–34710, 2019, doi: 10.1109/ACCESS.2019.2895092.

[55] 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 Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, Dec. 2018, doi: 10.1109/TNSM.2018.2861741.

[56] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE Local Computer Network Conference*, 2010, pp. 408–415. doi: 10.1109/LCN.2010.5735752.

[57] M. Ramadas, S. Ostermann, and B. Tjaden, “Detecting Anomalous Network Traffic with Self-organizing Maps.”

[58] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Perez-Diaz, “SDN-based architecture for transport and application layer DDoS attack detection by using machine and deep learning,” *IEEE Access*, vol. 9, pp. 108495–108512, 2021, doi: 10.1109/ACCESS.2021.3101650.

[59] R. Santos, D. Souza, W. Santo, A. Ribeiro, and E. Moreno, “Machine learning algorithms to detect DDoS attacks in SDN,” in *Concurrency and Computation: Practice and Experience*, John Wiley and Sons Ltd, Aug. 2020. doi: 10.1002/cpe.5402.

[60] A. Devarakonda, N. Sharma, P. Saha, and S. Ramya, “Network intrusion detection: A comparative study of four classifiers using the NSL-KDD and KDD’99 datasets,” in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jan. 2022. doi: 10.1088/1742-6596/2161/1/012043.

[61] 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,” in *2016 IEEE 13th international conference on e-business engineering (ICEBE)*, IEEE, 2016, pp. 56–63.

[62] S. Gupta and D. Grover, “A Comprehensive Review on Detection of DDoS Attacks using ML in SDN Environment,” in *Proceedings - International Conference on Artificial Intelligence and Smart Systems, ICAIS 2021*, Institute of Electrical and Electronics Engineers Inc., Mar. 2021, pp. 1158–1163. doi: 10.1109/ICAIS50930.2021.9395987.

[63] N. Bawany, J. Shamsi, and K. Salah, “DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions,” *Arab J Sci Eng*, vol. 42, Oct. 2017, doi: 10.1007/s13369-017-2414-5.

[64] R. T. Kokila, S. T. Selvi, and K. Govindarajan, “DDoS detection and analysis in SDN-based environment using support vector machine classifier,” in *2014 sixth international conference on advanced computing (ICoAC)*, IEEE, 2014, pp. 205–210.

[65] C. Li *et al.*, “Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN,” *International Journal of Communication Systems*, vol. 31, no. 5, p. e3497, 2018.

[66] 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,” in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, IEEE, 2018, pp. 202–206.

[67] Kali, “Hping3”.

[68] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, “Advanced study of SDN/OpenFlow controllers,” in *ACM International Conference Proceeding Series*, 2013. doi: 10.1145/2556610.2556621.

[69] http://mininet.org/, “mininet.”

[70] OVS, “openvswitch.”

[71] M. Alabadi, A. Habbal, and X. Wei, “Industrial internet of things: Requirements, architecture, challenges, and future research directions,” *IEEE Access*, 2022.

[72] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in *2010 IEEE symposium on security and privacy*, IEEE, 2010, pp. 305–316.

[73] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in *2010 IEEE symposium on security and privacy*, IEEE, 2010, pp. 305–316.

[74] R. Panigrahi and S. Borah, “A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems,” *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.

[75] N. N. Tuan, P. H. Hung, N. D. Nghia, N. Van Tho, T. Van Phan, and N. H. Thanh, “A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN,” *Electronics (Switzerland)*, vol. 9, no. 3, Mar. 2020, doi: 10.3390/electronics9030413.

[76] N. Moustafa and J. Slay, “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set,” *Information Security Journal: A Global Perspective*, vol. 25, no. 1–3, pp. 18–31, 2016.

[77] R. S. Olson and J. H. Moore, “TPOT: A tree-based pipeline optimization tool for automating machine learning,” in *Workshop on automatic machine learning*, PMLR, 2016, pp. 66–74.

[78] R. Durgut, Y. Y. Baydilli, and M. E. Aydin, “Feature selection with artificial bee colony algorithms for classifying Parkinson’s diseases,” in *Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference: Proceedings of the EANN 2020 21*, Springer, 2020, pp. 338–351.

[79] P. R. K. Varma, V. V. Kumari, and S. S. Kumar, “Feature selection using relative fuzzy entropy and ant colony optimization applied to real-time intrusion detection system,” *Procedia Comput Sci*, vol. 85, pp. 503–510, 2016.

[80] J. Zhang and M. Zulkernine, “Network Intrusion Detection using Random Forests.,” in *Pst*, Citeseer, 2005.

[81] M. Dash and H. Liu, "Feature selection for classification," *Intelligent data analysis*, vol. 1, no. 1–4, pp. 131–156, 1997.

[82] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, pp. 197–227, 2016.

[83] Y. Baydilli and U. Atila, "Understanding effects of hyper-parameters on learning: A comparative analysis," in *Proceedings of the International Conference on Advanced Technologies, Computer Engineering and Science, Safranbolu, Turkey*, 2018, pp. 11–13.

[84] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[85] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[86] N. Chand, P. Mishra, C. R. Krishna, E. S. Pilli, and M. C. Govil, "A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection," in *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring)*, IEEE, 2016, pp. 1–6.

[87] K. Korjus, M. N. Hebart, and R. Vicente, "An efficient data partitioning to improve classification performance while keeping parameters interpretable," *PLoS One*, vol. 11, no. 8, p. e0161788, 2016.

[88] İ. Tozlu, Ş. G. Öğüdücü, A. Çelebi, S. Kalaycı, and S. Arslan, "Accelerating balance sheet adjustment process in commercial loan applications with machine learning methods," in *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, IEEE, 2019, pp. 1–8.

[89] Y. Y. Baydilli and Ü. Atila, "Classification of white blood cells using capsule networks," *Computerized Medical Imaging and Graphics*, vol. 80, p. 101699, 2020.

[90] S. R. Sharma, B. Singh, and M. Kaur, "A novel approach of ensemble methods using the stacked generalization for high-dimensional datasets," *IETE J Res*, pp. 1–16, 2022.

[91] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[92] O. Dakkak, A. S. Che Mohamed Arif, and S. Awang Nor, "A critical analysis of simulators in grid," *J Teknol*, vol. 77, no. 4, pp. 111–117, 2015.

[93] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert, "Deep learning-based electroencephalography analysis: a systematic review," *J Neural Eng*, vol. 16, no. 5, p. 051001, 2019.

[94] I. Stancin and A. Jovic, "An overview and comparison of free Python libraries for data mining and big data analysis. 42nd International Convention on Information and Communication Technology," *Electronics and Microelectronics. doi*, vol. 10, 2019.

[95] M. J. J. Douglass, "Book Review: Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow, by Aurélien Géron: O'Reilly Media, 2019, 600 pp., ISBN: 978-1-492-03264-9." Springer, 2020.

[96] Y. Y. Baydilli, U. Atila, and A. Elen, "Learn from one data set to classify all– A multi-target domain adaptation approach for white blood cell classification," *Comput Methods Programs Biomed*, vol. 196, p. 105645, 2020.

[97] A. K. Sarica and P. Angin, "Explainable security in SDN-based IoT networks," *Sensors*, vol. 20, no. 24, p. 7326, 2020.

[98] S. Zian, S. A. Kareem, and K. D. Varathan, "An empirical evaluation of stacked ensembles with different meta-learners in imbalanced classification," *IEEE Access*, vol. 9, pp. 87434–87452, 2021.

[99] A. Ahmad, E. Harjula, M. Ylianttila, and I. Ahmad, "Evaluation of machine learning techniques for security in SDN," in *2020 IEEE Globecom Workshops (GC Wkshps*, IEEE, 2020, pp. 1–6.

[100] O. Dakkak, Y. Fazea, S. A. Nor, and S. Arif, "Towards accommodating deadline driven jobs on high performance computing platforms in grid computing environment," *J Comput Sci*, vol. 54, p. 101439, 2021.

[101] K. Hasan, S. Shetty, A. Hassanzadeh, and M. Ben Salem, "Self-healing cyber resilient framework for software defined networking-enabled energy delivery system," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2018, pp. 1692–1697.

[102] P. Thorat, S. M. Raza, D. T. Nguyen, G. Im, H. Choo, and D. S. Kim, "Optimized self-healing framework for software defined networks," in *Proceedings of the 9th International Conference on Ubiquitous Information Management and Communication*, 2015, pp. 1–6.

[103] L. Ochoa-Aday, C. Cervelló-Pastor, and A. Fernández-Fernández, "Self-healing and SDN: bridging the gap," *Digital Communications and Networks*, vol. 6, no. 3, pp. 354–368, 2020.

[104] F. lahmood HAMEED and O. DAKKAK, "Brain Tumor Detection and Classification Using Convolutional Neural Network (CNN)," in *2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, IEEE, 2022, pp. 1–7.

[105] O. Dakkak, S. Arif, and S. A. Nor, "Resource allocation mechanisms in computational grid: A survey," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, pp. 6662–6667, 2006.

[106] M. Belyaev and S. Gaivoronski, "Towards load balancing in SDN-networks during DDoS-attacks," in *2014 international science and technology conference (modern networking technologies)(MoNeTeC)*, IEEE, 2014, pp. 1–6.

[107] Y. Zhou, K. Zheng, W. Ni, and R. P. Liu, "Elastic switch migration for control plane load balancing in SDN," *IEEE Access*, vol. 6, pp. 3909–3919, 2018.

[108] A. A. Abdelltif, E. Ahmed, A. T. Fong, A. Gani, and M. Imran, "SDN-based load balancing service for cloud servers," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 106–111, 2018.

[109] M. N. Hindia, A. W. Reza, O. Dakkak, S. Awang Nor, and K. A. Noordin, "Cloud computing applications and platforms: A Survey," 2014.

[110] J. Son and R. Buyya, "A taxonomy of software-defined networking (SDN)-enabled cloud computing," *ACM computing surveys (CSUR)*, vol. 51, no. 3, pp. 1–36, 2018.

[111] M. Banikazemi, D. Olshefski, A. Shaikh, J. Tracey, and G. Wang, "Meridian: an SDN platform for cloud network services," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 120–127, 2013.

[112] X. Wu, L. Xiao, Y. Sun, J. Zhang, T. Ma, and L. He, "A survey of human-in-the-loop for machine learning," *Future Generation Computer Systems*, 2022.

[113] S. Aiman, S. Hassan, A. Habbal, A. Rosli, and A. H. M. Shabli, "Smart electricity billing system using blockchain technology," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 10, no. 2–4, pp. 91–94, 2018.

[114] S. M. Karim, A. Habbal, S. A. Chaudhry, and A. Irshad, "BSDCE-IoV: Blockchain-Based Secure Data Collection and Exchange Scheme for IoV in 5G Environment," *IEEE Access*, 2023.

[115] H. Seyam and A. Habbal, "A Systematic Review of Blockchain-based Identity Management Solutions," in *International Conference on Recent Academic Studies*, 2023, pp. 246–253.

[116] M. Raeisi-Varzaneh, O. Dakkak, A. Habbal, and B.-S. Kim, "Resource Scheduling in Edge Computing: Architecture, Taxonomy, Open Issues and Future Research Directions," *IEEE Access*, vol. 11, pp. 25329–25350, 2023.

RESUME

Tasnim Kamal ALASALI is an accomplished individual with a Bachelor's Degree in Computer Engineering from the Syrian Private University. She graduated in 2016 and demonstrated a strong understanding of complex technological concepts during her studies. Tasnim furthered her education by pursuing a Master's Degree in Computer Engineering at Karabük University from 2021 to 2023. Her postgraduate studies allowed her to delve into advanced topics and conduct extensive research, fueling her passion for the field. Tasnim's solid educational foundation and commitment to professional growth make her a valuable asset in the realm of computer engineering.