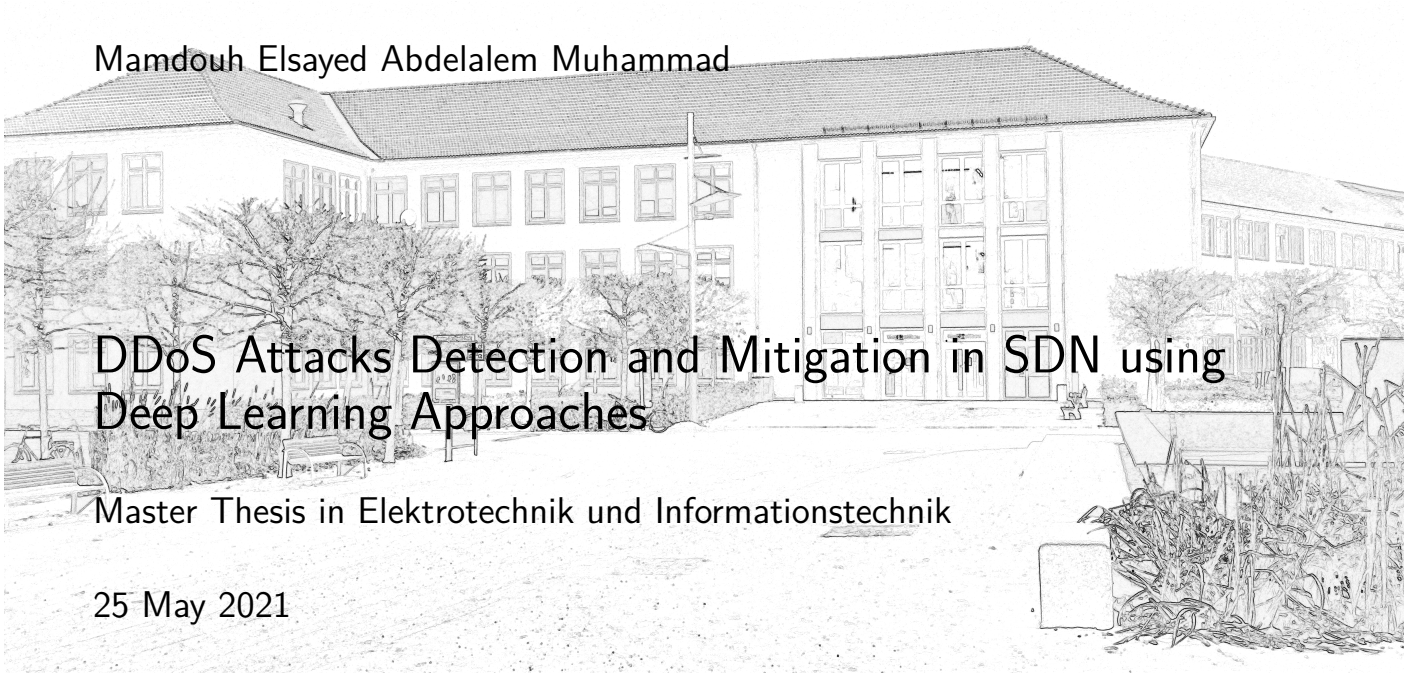
TECHNISCHE UNIVERSITÄT
ILMENAU

Communication Networks Group

Mamdouh Elsayed Abdelalem Muhammad



DDoS Attacks Detection and Mitigation in SDN using Deep Learning Approaches

Master Thesis in Elektrotechnik und Informationstechnik

25 May 2021

Please cite as:
Mamdouh Elsayed Abdelalem Muhammad, "DDoS Attacks Detection and Mitigation in SDN using Deep Learning Approaches," Master Thesis (Masterarbeit), Technische Universität Ilmenau, Deptpartment of Electrical Engineering and Information Technology, May 2021.

Technische Universität Ilmenau
Department of Electrical Engineering and Information Technology
Communication Networks Group

Helmholtzplatz 2 · 98693 Ilmenau · Germany
http://www.tu-ilmenau.de/kn

DDoS Attacks Detection and Mitigation in SDN using Deep Learning Approaches

Master Thesis in Elektrotechnik und Informationstechnik

submitted by

Mamdouh Elsayed Abdelalem Muhammad

born on 26. January 1991
in Monofiya

in the

Communication Networks Group

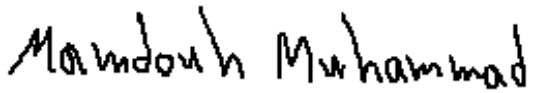
Deptartment of Electrical Engineering
and Information Technology
Technische Universität Ilmenau

Advisors: **Prof. Dr. rer. nat. J. Seitz**
M.Sc. Abdullah Soliman Alshra'a

Submission Date: **25 May 2021**

Declaration

I declare that the work is entirely my own and was produced with no assistance from third parties.
I certify that the work has not been submitted in the same or any similar form for assessment to any other examining body and all references, direct and indirect, are indicated as such and have been cited accordingly.



(Mamdouh Elsayed Abdelalem Muhammad)
Ilmenau, 25 May 2021

Acknowledgement

"I would like to express my genuine thanks to my advisor M.Sc. ***Abdullah Soliman Alshra'a*** *for his permanent support, without whom I would not have completed this study. His deep expertise and limitless guidance were the prime movers in inspiring me throughout the whole time.*

My heartfelt gratitude to Prof. ***Jochen Seitz*** *for allowing me to complete my thesis in the Communication Networks Group. He has always been an inspiration to me since he supports his students by sharing the knowledge and personally paving the way by every possible means to ease the journey.*

I dedicate this work first to God Almighty, my Mom & Dad, who were always there for me, my sister ***Aya****, and my brother* ***Ahmed****.*

Last but not least, my sincere gratitude to my friend ***Aly Hosny*** *and other friends of TU Ilmenau that have been there for me."*

Abstract

Software-Defined Networking (SDN) is a promising approach to developing traditional networks by separating the control plane from the data plane. Due to its new unique concepts and features of having a centralized point, i.e., the SDN controller, to create, manage and troubleshoot the networks, SDN requires more analyses to guarantee its viability from a security point of view. The concept of having a centralized view is a blessing that may turn into a curse. Targeting the SDN controller availability through launching Distributed Denial of Service (DDoS) attacks is one of the most crucial and critical challenges that exist in SDN. DDoS attacks are famous cyber-attacks that focus on resource depletion and causing network services unavailability. The objective of this study is to simulate the DDoS attacks on SDN network and test the effectiveness of using deep learning models to detect and mitigate such attacks in a SDN multi-controller environment. For accurate handling and analyzing the DDoS attacks and emulating the actual physical networking environments, the CICDDoS2019 dataset is chosen. Experimental results declared that the DDoS attack could be detected and mitigated to prevent the unavailability of the SDN controller. In addition, The proposed method outperforms the traditional machine learning methods.

Kurzfassung

SDN ist ein vielversprechender Ansatz zur Entwicklung traditioneller Netzwerke durch die Trennung der Steuerungsebene von der Datenebene. Aufgrund seiner neuen, einzigartigen Konzepte und Funktionen, bei denen ein zentraler Punkt, d. h. der SDN-Controller, für die Erstellung, Verwaltung und Fehlerbehebung der Netzwerke zuständig ist, erfordert SDN mehr Analysen, um seine Durchführbarkeit aus Sicherheitsgesichtspunkten zu gewährleisten. Das Konzept einer zentralisierten Ansicht ist ein Segen, der sich in einen Fluch verwandeln kann. Die Verfügbarkeit des SDN-Controllers durch das Starten von DDoS-Angriffen ins Visier zu nehmen, ist eine der wichtigsten und kritischsten Herausforderungen, die in SDN existieren. DDoS-Angriffe sind bekannte Cyber-Attacken, die sich auf die Erschöpfung von Ressourcen konzentrieren und die Nichtverfügbarkeit von Netzwerkdiensten verursachen. Das Ziel dieser Studie ist es, die DDoS-Angriffe auf das SDN-Netzwerk zu simulieren und die Effektivität der Verwendung von Deep-Learning-Modellen zur Erkennung und Entschärfung solcher Angriffe in einer SDN-Umgebung mit mehreren Controllern zu testen. Um die DDoS-Angriffe genau zu behandeln und zu analysieren und die tatsächlichen physischen Netzwerkumgebungen zu emulieren, wird der CICDDoS2019-Datensatz gewählt. Die experimentellen Ergebnisse erklärten, dass der DDoS-Angriff erkannt und abgeschwächt werden konnte, um die Nichtverfügbarkeit des SDN-Controllers zu verhindern. Darüber hinaus übertrifft die vorgeschlagene Methode die traditionellen Methoden des maschinellen Lernens.

Contents

Chapter 1

Introduction

Due to the rapid changes in network traffic flow, traditional networks must be innovated. For example, new technologies like Edge-Computing, Internet of Things (IoT) , blockchain and Artificial Intelligence (AI) needs a new creative and innovative approaches.
Based on Cisco annual internet report [1], by 2023, almost two-thirds of the world's population will have access to the Internet. In addition, there will be 5.3 billion active Internet users compared to 3.9 billions in 2018.
Internet users expect to low delay or even delay-free connections and although devices and applications, have all seen significant advancements, the network traffic engineering has remained unchanged. Ossification of the Internet is a challenge for network engineers and managers as the complexity of the internet physical infrastructure is increasing [2].
SDN is a modern approach to improve network management and service [3]. It has opened up many ways for network revolution with the aid of its basic concept of decoupling the control plane and data plane. Besides being efficient in controlling and managing the network from a centralized point, SDN help in applying network programmability [3] [4] .
In addition, SDN aims to save costs by customizing the hardware as needed instead of using vendor-based over-featured costly devices [5]. SDN's main idea is to have a centralized point, i.e., the SDN controller, by which we can configure, automate and troubleshoot our network. Furthermore, SDN allows us to alter the network instinctively by adding, adapting , and removing networks commensurating with our needs.

But the main challenge in this approach is that the controller is a Single Point Of Failure (SPOF), and by affecting its availability, this accelerates the collapse of the network [6]. So despite its efficiency and simplicity, SDN suffers from a significant drawback, which is the lack of security for the SDN controller.

1.1 SDN versus Traditional Networks

As depicted in Figure 1.1, the main difference between SDN and traditional networks is in the structure. SDN allows for unified control of data plane forwarding devices regardless of the technologies used to link them, as well as a central network view of all data paths.

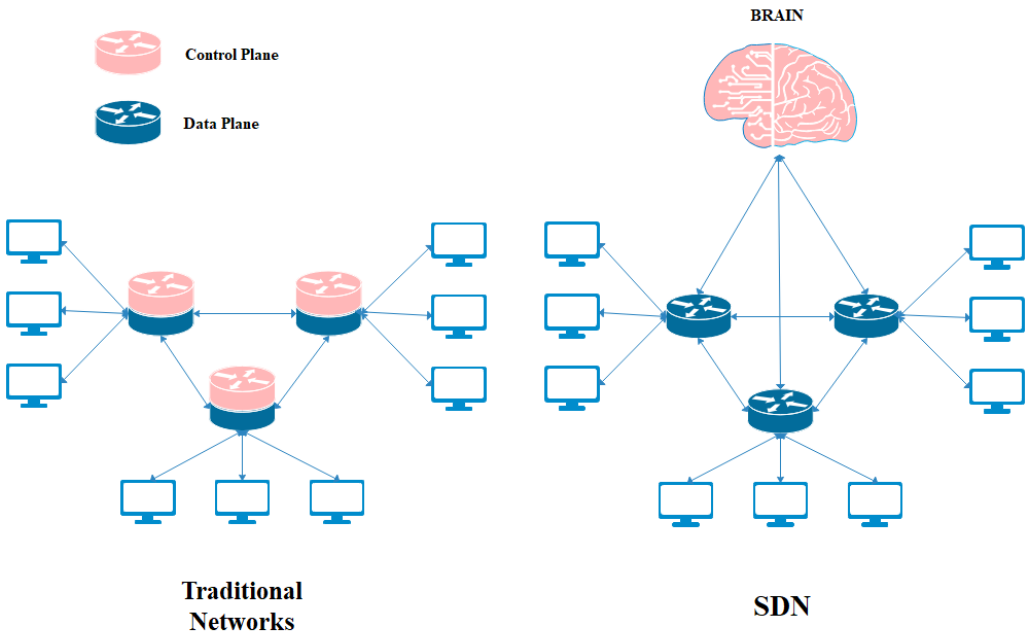


Figure 1.1 – Traditional networks vs SDN networks

In traditional networks, each switch is a standalone entity, it can process the traffic and independently take forwarding decisions. This is due the fact the each switch in the traditional networks contains both the control plane and data plane as can be clearly seen from the figure above. In opposite, SDN shifts from a vertical to a horizontal integration of network elements, with distinct separate working layers for policy specification, compliance, and execution [6]. This capability enables network managers to dynamically change traffic in response to changing requirements. Additionally, SDN enables the programmers to write software programs without the need to know the actual workflow of SDN protocols. The reason behind that is that the programmers will deal with the API of the controller [3].

1.2 Benefits of Software Defined Networking (SDN)

Having a single centralized managing point offers many advantages in SDN over traditional networks. The main benefits of SDN are the following:

- **Simplification [7] [8]:** As mentioned above, internet ossification is a major challenge. We are in desperate need of intelligence in managing the internet networks. As Reduced Instruction Set Computer (RISC) model is considered an evolution of Complex Instruction Set Computing (CISC) model and instead of focusing on th hardware as in CISC, we tend now to focus on software as RISC. Also, we can consider SDN as the reset for the typical traditional network design.
- **Programmability [3]:** SDN provides efficient environment for programmers using automation tools like Chef and Puppet to orchestrate and automate the SDN network [3]. This is due to the existence of single centralized point i.e. the SDN controller. Using Application Programming Interface (API)s such as Representational state transfer (Rest) API provides a common interface to program and develop applications.
- **Reduction of costs [7], [5]:** In the era of virtualization, we need standard commodity hardware. Original Device Manufacturer (ODM) manufacture the so-called white boxes, which are unbranded devices. White boxes tendency is booming due its cheap price. SDN support using the white box switches.
- **Elasticity [9]:** Running the SDN controller as a software program enables the controller to easily adjust its resource usage based on the necessary capability. Resource allocations can be used to increase the SDN controller capabilities.
- **Protocol Independence [9]:** Instead of using over-featured switches running proprietary protocols, in SDN, protocol choosing is based on your need.

Chapter 2

Software Defined Networking (SDN)

SDN first Forwarding and Control Element Separation (ForCES) architecture standardisation done by Internet Engineering Task Force (IETF) in 2003 in RFC3654 [10] and confirmed in 2004 in RFC3746 [11] [12]. According to [13], there are three layers in SDN. As as depicted in Figure 2.1. Firstly, we have the the Infrastructure layer (Data Plane) which contains packet forwarding dummy devices that take actions based on rules sent by the control plane. Secondly, we have the control plane, which is the brain of the network. It represent the intelligence part of the network because it contains the controller which control and orchestrate the network by providing the data plane with actions needed to forward the packets. The controller is doing that based also on the instructions from the application layer. Lastly, the applications layer is the layer that contains the user applications to interact and control the control plane. The detail of these layers is summarized in following sections.

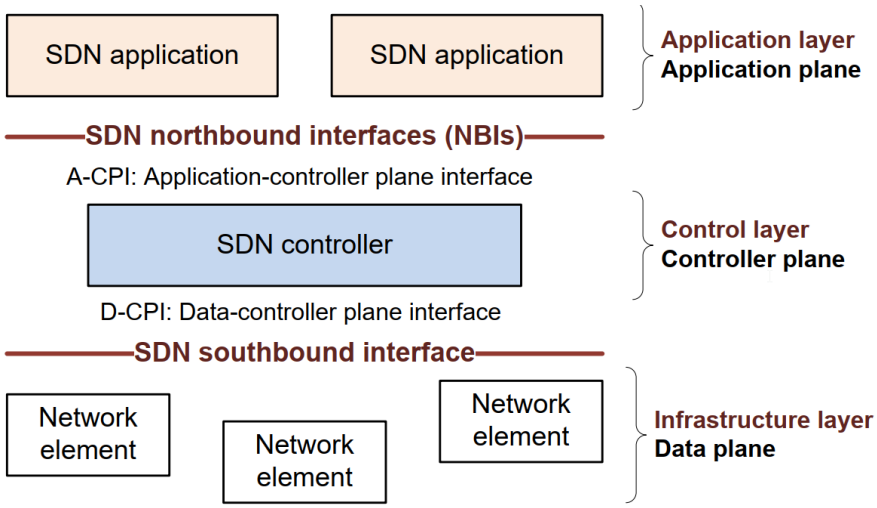


Figure 2.1 – SDN Basic components

2.1 SDN Components

2.1.1 Infrastructure Layer

The Infrastructure layer/Data Plane is the lowest Layer in the SDN structure. It is responsible for buffering, scheduling and forwarding the data. If the data is new to the SDN switch and there are no flow entries in the forwarding table that matches this data, it will consult the control plane. Based on the response of the control plane, the data plane is forwarding or dropping the data. If the data already exist in the forwarding table, the data plane will take actions without the support of the other two layers. Many researchers proposed ideas about data plane programmability to support new protocols and telemetry [14] [15]. Others proposed to choose a specific metric to evaluate the programmability and the flexibility of the data plane to increase them in level of packet-switching [16]. Two SDN switches types are offered, physical and virtual. Physical switches like Pronto 3290, NEC PF5240, Pica8 P-3290 and Hewlett-Packard 8200zl. Virtual switches like OpenFaucet, OpenFlowJ, Nettle and Pica8 [17].

2.1.2 Control Plane

The control plane is the brain of the network and acts like a bridge between the network manager and the forwarding devices. It pushes the configurations defined by the network manager in the application layer to the data plane. In addition, it contains the SDN controllers. In table Table 2.1 [18], we can see information about the best known SDN controllers. As depicted in Figure 2.2, one of the prominent factors in determining which controllers is suitable for my network is whether the controller has distributed or centralized architecture [6]. There will be only one controller in the centralized architecture, and it is solely responsible for switches control. In the logically centralized architecture, there will be set of controllers. These controllers will manage the whole network as all of them have the same network view and the same information. This helps to get better performance and avoid the problems e.g. SPOF and scalability that exist in the centralized architecture [19]. Logically distributed architecture means that each controller will manage specific domain of devices not the whole network [20]. Logically distributed controllers have East-West bound API that allow them to exchange Information [6].

Controller	Programming Language	Further development
Ryu	Python	Yes
POX	Python	No
Floodlight	Java	Yes
OpenDaylight	Java	Yes
Beacon	Java	No
ONOS	Java	Yes
NOX	C ++, Python	No
Trema	Ruby, C	Yes

Table 2.1 – Famous SDN controllers

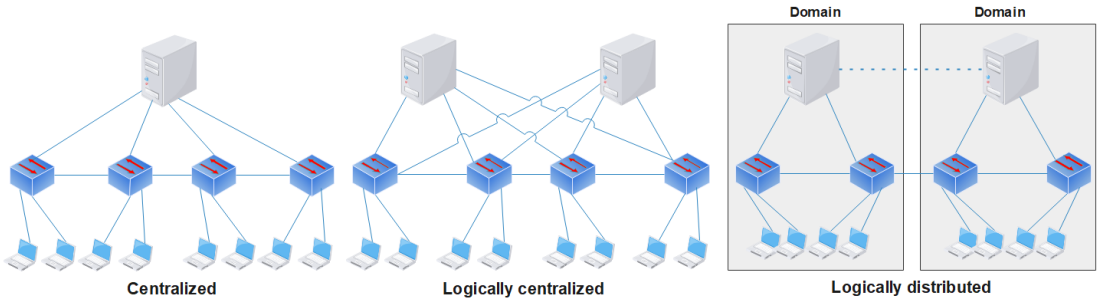


Figure 2.2 – SDN Topologies Architectures

2.1.3 Application Plane

Application layer is the top layer in SDN structure as depicted in Figure 2.1. It contains the applications written by computer programmers. The network manager can use them in monitoring and controlling the network. Moreover, visualization and security applications interacts with control plane to retrieve all needed information for network management.

2.1.4 SDN APIs

SDN offers the support of many programmable interfaces. This facilitate the communication between the three layers as shown in Figure 2.1. The best known important APIs for SDN are Northbound API and Southbound API described by [21] are discussed below.

- **Southbound API: [6] [5]** As shown in Figure 2.1, Southbound API is located between the controller and the data plane devices. Its role is to provide the controller with needed interface to interact with the data plane layer. The controller is using it to add, delete or modify flow entries in the SDN switches. Furthermore, the controller is using it to retrieve any statistics regarding device, ports and links. Various Southbound protocols are available, e.g. Simple Network Management Protocol (SNMP), Border Gateway Protocol (BGP) [21], Locator/ID Separation Protocol (LISP) [22] and OpenFlow [23]

- **East-West bound API**: Control panel scalability is one of the major benefits of the logically distributed and logically centralized SDN architectures. East-West bound API is the interface for the logically distributed control plane that allow data transition and exchange between the SDN controllers. Until now, there is no standard east-west interface [20]. However, many works proposed many interface such as SDNi [24]. SDNi is a east-west protocol that contains different types of messages, e.g. Flow setup/tear-down/update and Reachability update [24].

- **Northbound API: [6]** As shown in Figure 2.1, Northbound API is a software ecosystem that provides a standardized interface for application development. It is the interface between the controller and the application plane. So in the need of a specific application to measure Quality Of Service (QoS), install load-balancing solution or force specific security settings, Northbound API is used by the user to interact with the controller. This is done by using the Northbound API [5]. On the contrary to the Southbound, Northbound API suffers from absence of standardization. Only some controllers offers customized Northbound API.

2.2 OpenFlow Protocol

As mentioned in section Section 2.1.4, the controller needs a Southbound API to interact with the data plane. For southbound APIs, OpenFlow is the most widely used protocol [6]. The SDN controller uses the OpenFlow protocol in guiding the OpenFlow-enabled switches on how to process packets [6]. Initially, the OpenFlow protocol was created for use on a campus network [25]. Later, vendors became interested and invested in it.

2.2.1 Overview

As depicted in Figure 2.3 [26], the OpenFlow protocol is the middleware between the SDN controller and the OpenFlow switch. Proactivly or reactivly can the controller add, edit or remove any flow entry in the flow table [26]. The flow of the data in the OpenFlow switch is depicted in Figure 2.4 [27].

2.2.2 OpenFlow Messages

The OpenFlow protocol supports three main messages types, controller-to-switch, asynchronous, and symmetric. Each message type contains several sub-types [26]. Controller-to-switch messages are originated from the controller to the switch and are used to manage or guide the switch. Asynchronous messages are change-dependent messages that are triggered by the switch to inform the controller about new network event any any change happened to

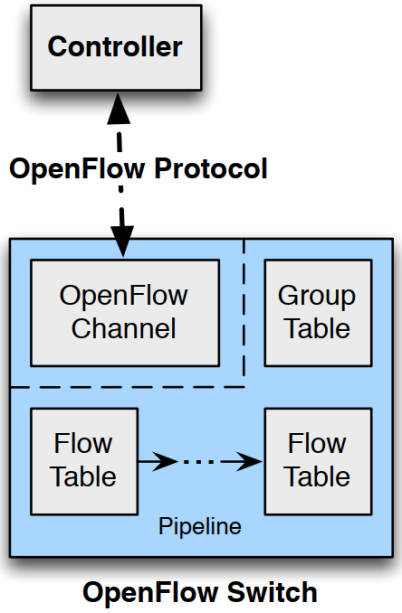


Figure 2.3 – OpenFlow switch internal components

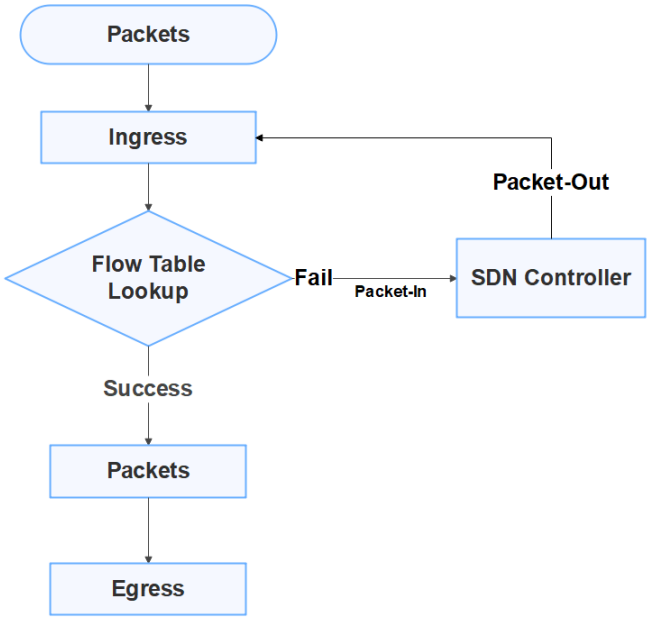


Figure 2.4 – OpenFlow switch flow processing flowchart

the switch state. Symmetric messages are sent either by the controller or the switch without no request needed.

Message Type	Message Name	Purpose
Controller-to-Switch	OFPT_FEATURES_REQUEST	Identify the switch and retrieve its features.
	OFPT_MULTIPART_REQUEST	Used in case of messages that can't be sent in one message to encode the large requests and reply messages.
	OFPMP_FLOW (request)	Request info. about flow entries.
	OFPT_PACKET_OUT	sending out a packet through the datapath.
	OFPMP_GROUP	Get information about groups.
	OFPT_FLOW_MOD	To adjust the flow table.
	OFPT_GROUP_MOD	To make changes to a the group table.
	OFPT_SET_CONFIG OFPT_GET_CONFIG_REQUEST	The switch's configuration parameters can be set and retrieved.
	OFPT_TABLE_MOD	To configure a switch's dynamic state.
	OFPT_PORT_MOD	To change the properties of a port.
	OFPT_METER_MOD	To change the value of a meter and a virtual meter.
	OFPMP_DESC (Request)	To get information about the switch (such as the manufacturer, software version, and serial number).
	OFPMP_AGGREGATE (Request)	To aggregate information from several flow entries.
	OFPMP_TABLE (Request)	To learn the most important information about the tables.
	OFPMP_TABLE_DESC (Request)	To retrieve a switch's current table configuration.

Message Type	Message Name	Purpose
Controller-to-Switch	FPMP_QUEUE_STATS (Request)	To obtain information on one or more ports' queues.
	OFPMP_GROUP (Request)	To obtain information about one or more groups.
	OFPMP_PORT_DESCRIPTION (Request)	To learn about all ports that support OpenFlow.
	OFPMT_METER (Request)	To obtain meter information.
	OFPT_BARRIER_REQUEST	To inquire about the operations that have been completed.
	OFPT_ROLE_REQUEST	The controller uses this to adjust its role.
Asynchronous messages	OFPT_ROLE_STATUS	The switch uses this to adjust the controller role.
	OFPT_TABLE_STATUS	To notify the controller when a table's status changes.
	OFPT_PORT_STATUS	By the switch to notify the controller if any port is added, modified or removed.
	OFPT_PACKET_IN	By the switch to notify the controller if packet received.
	OFPT_FLOW_REMOVED	The datapath will notify the controller about deletion or timing out of a flow if the controller requested that.
	OFPT_ERROR_MSG	Used by either the controller or the switch to notify each other about problems.
Symmetric Messages	OFPT_HELLO	To negotiate about the supported version of OpenFlow between the controller and the switch.
	OFPT_ECHO_REQUEST OFPT_ECHO_Reply	To act as keep alive between the controller and the switch and helping in measuring the latency and the bandwidth.

2.2.3 OpenFlow Tables

The components of flow tables and group tables, as well as the fundamentals of matching and action handling, are covered in this section. For an overall overview of the OpenFlow switch tables components, this is depicted in Figure 2.5.

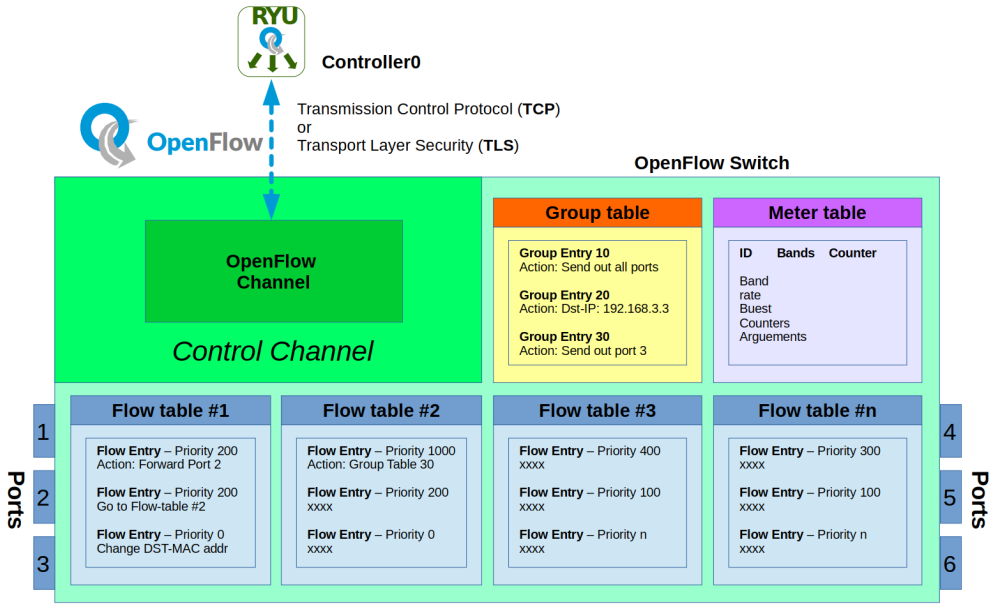


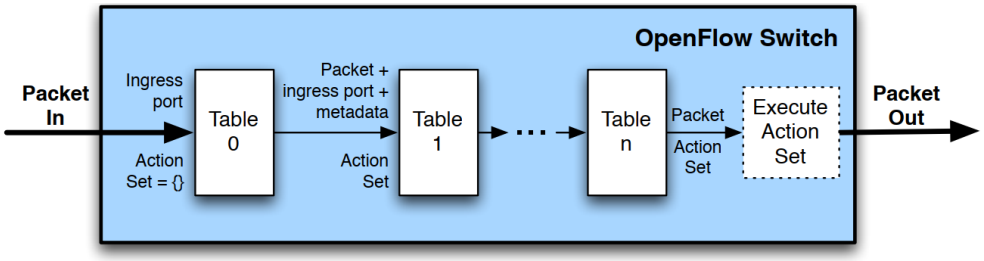
Figure 2.5 – OpenFlow Switch Internal components [28]

2.2.3.1 Pipeline Processing

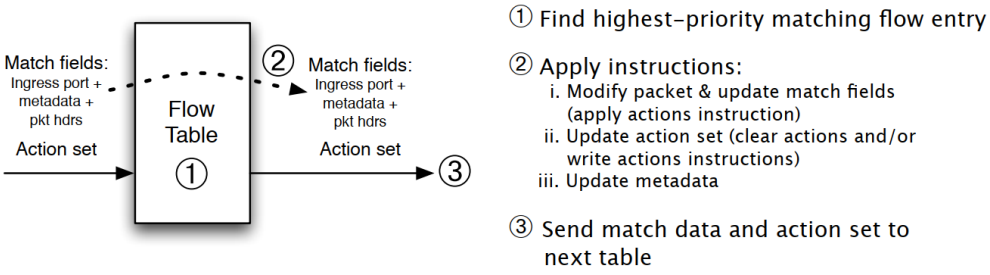
There are two kinds of OpenFlow-compliant switches. The first is **OpenFlow-hybrid** switches which support the two pipelines, the OpenFlow pipelines [26] and also the traditional Ethernet switching pipelines. The traditional Ethernet switching processes are like L2 switching , L3 routing, Virtual Local Area Network (VLAN) tagging, QoS and Access Control List (ACL). The second is **OpenFlow-only** switches which support the OpenFlow pipeline only.
As shown in Figure 2.6, The processing sequence in OpenFlow switch. This pipeline describes how the flow tables is dealing with packets.

2.2.3.2 Flow Tables

An OpenFlow switch consists of at least one flow table. Flow tables enable the switch to look the packets up and forward them in case of matching. In addition, Flow tables contains the policies and rules for fast packet processing [29]. The following tables show the change in the flow table entry fields. In OpenFlow 1 [30], OpenFlow 1.2 [31], OpenFlow 1.3 [32], OpenFlow



(a) Packets are matched against multiple tables in the pipeline



(b) Per-table packet processing

Figure 2.6 – SDN Pipeline Process

1.3.5 [26] and OpenFlow 1.5 [33], we can see the changes in the flow entry fields in the flow table.

Header Fields	Counters	Actions

Table 2.2 – Flow table fields in OpenFlow version 1.0.0 [30]

Header Fields	Counters	Instructions

Table 2.3 – Flow table fields in OpenFlow version 1.2 [31]

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Table 2.4 – Flow table fields in OpenFlow version 1.3.0 [32]

- **Match Fields**: Supported matching field to match the packet. In Table 2.2.3.2, we can see the different types of match fields in many OpenFlow versions [34]. In Figure 2.7, we can see the content of the matching field in the OpenFlow protocol [5].

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Table 2.5 – Flow table fields in OpenFlow version 1.3.5 [26]

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie	Flags

Table 2.6 – Flow table fields in OpenFlow version 1.5.1 [33]

Field	**OF1.0**	**OF1.1**	**OF1.2**	**OF 1.3&1.4**
Ingress Port	X	X	X	X
Metadata		X	X	X
Ethernet: src, dst, type	X	X	X	X
IPv4: src, dst, proto, ToS	X	X	X	X
TCP/UDP: src port, dst port	X	X	X	X
MPLS: label, traffic class		X	X	X
OXM			X	X
IPv6			X	X
IPv6 Extension Headers				X

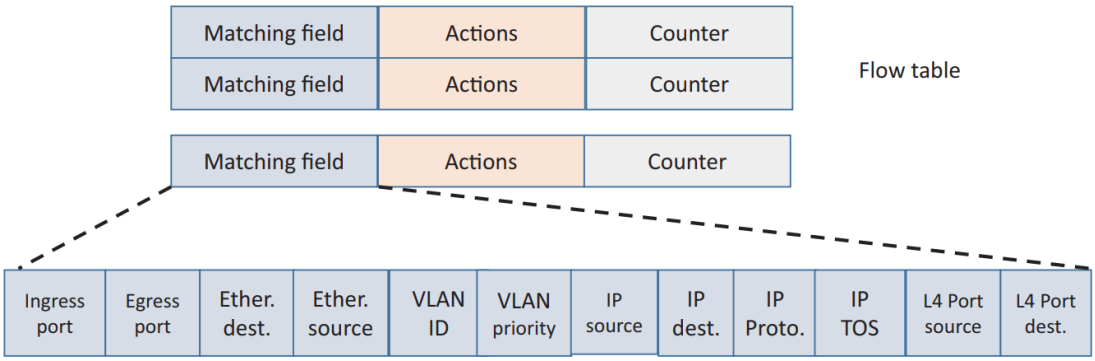


Figure 2.7 – SDN Matching Fields

- **Priority** : Determine the priority of the flow entry.
- **Counters**: Increased once a packet is processed.
- **Instructions**: To modify the pipeline processing or the action set.
- **Timeouts** : Maximum time before flow expiration by the switch.
- **Cookie**: A value used by the controller to filter flow entries.
- **Flags**: Management of the flows based on flag type.

2.2.3.3 Group Tables

For additional packet processing and further forwarding decisions, group tables are being used. In OpenFlow 1.2 [31], OpenFlow 1.3 [32], OpenFlow 1.3.5 [26] and OpenFlow 1.5.1 [33], we can see that the group entry structure in the group table consist of the structure as shown in Table 2.7

Group Identifier	Group Type	Counters	Action Buckets

Table 2.7 – Group table fields in OpenFlow version 1.3.5 [26]

- **Group Identifier**: To uniquely identify the group.
- **Group Type** : To identify the type of the group as there is required group types and optional group types.
- **Counters**:Increased once a packet is processed by a group.
- **Action Buckets**: A list of actions are stored in an action bucket.

2.2.3.4 Meter Table

Meter Table [26] is used in implementing QoS operations in OpenFlow and it contains meters for each flow. The meter table fields can be seen in Table 2.8. As shown in Figure 2.8, there may be a meter and attached to it are multiple bands [7].

Meter Identifier	Meter Bands	Counters

Table 2.8 – Meter table fields in OpenFlow version 1.3.5 [26]

2.2.4 OpenFlow Protocol Versions

2.2.4.1 OpenFlow v1.0

It is the most widely used OpenFlow version and was published in December, 2009. The matching fields in the header fields as shown in Table 2.2 contain 12 matching elements. Matching based on source address, destination address or Transmission Control Protocol (TCP)/User Datagram Protocol (UDP) source and destination port numbers is possible [34]. Figure 2.9 is showing the usage of OpenFlow version 1.0 as the communication channel between the SDN controller and the OpenFlow switch.

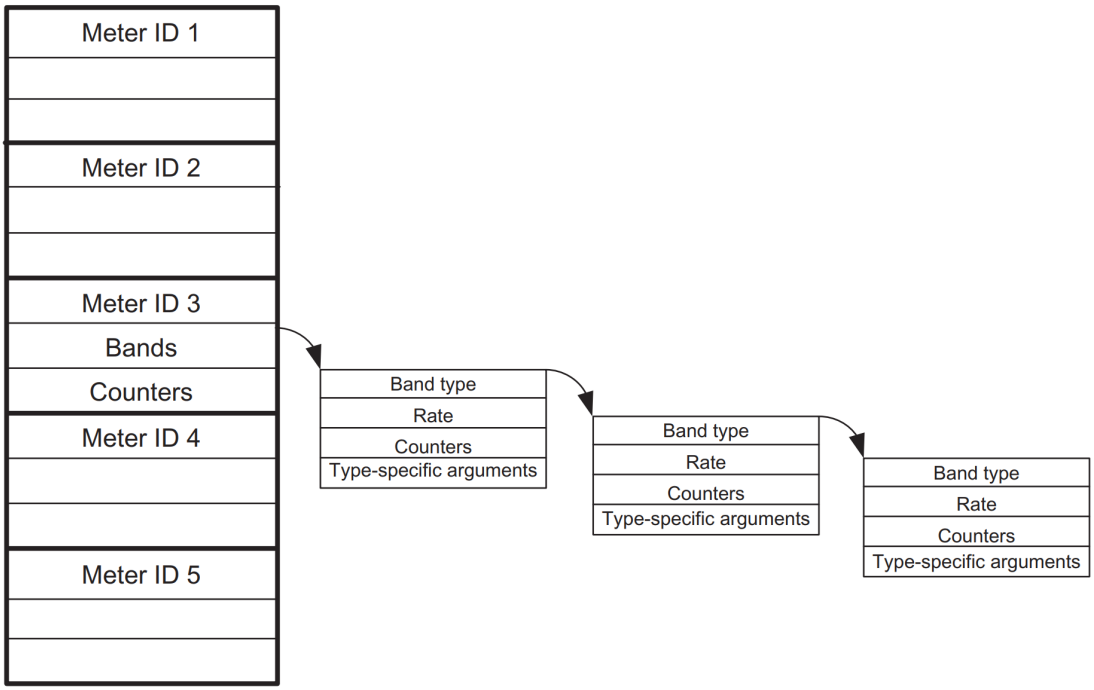


Figure 2.8 – Meter table components [7]

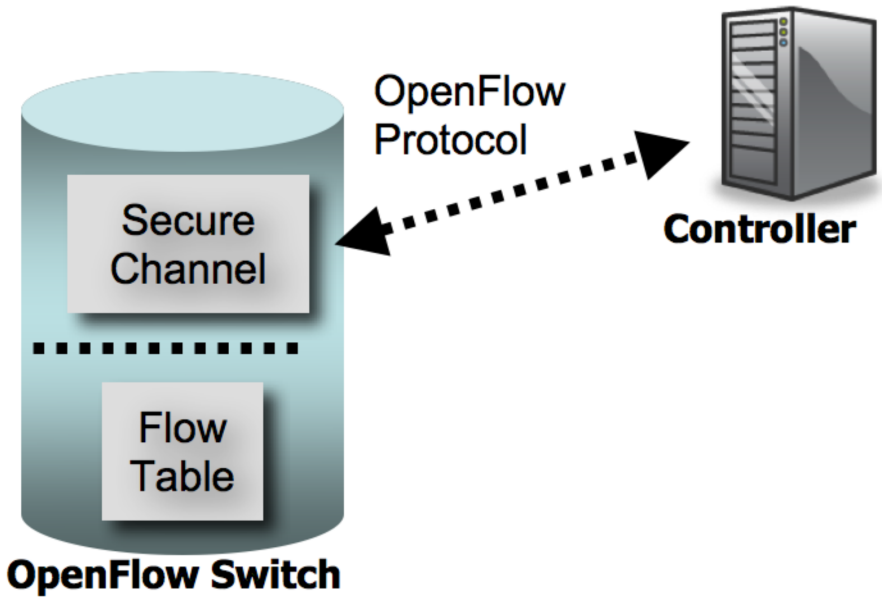


Figure 2.9 – Communication via OpenFlow v 1.0 in SDN [30]

2.2.4.2 OpenFlow v1.1

It was released in February, 2011. The main two new concepts introduced in OpenFlow v1.1 are multiple flow tables and the group table. In addition, matching packets in OpenFlow v1.1 is supporting Multiprotocol Label Switching (MPLS) and VLAN.

2.2.4.3 OpenFlow v1.2

It was released in December, 2011. Matching packets in OpenFlow v1.2 is supporting OpenFlow Extensible Match (OXM) and Internet Protocol version 6 (IPv6): src, dst, flow label, Internet Control Message Protocol version 6 (ICMPv6).

2.2.4.4 OpenFlow v1.3

It was released in April, 2012. Extension of QoS feature by adding the meter tables was significant improvement in OpenFlow v1.3 [35]. Furthermore, the concept of table-miss entry as an extension for the flow table was a second major improvement. In addition, the support for IPv6 Extension Headers as a matching field was introduced.

2.2.4.5 OpenFlow v1.4

It was released in August, 2013. The main two new concepts introduced by OpenFlow v1.4 [35] are the Synchronized table and the Bundle.

In 2.10, We can see the different types of statistics that exist in various OpenFlow tables. In Table 2.9 [35], is a full comparison between OpenFlow versions indicating the Major features, the reason and the use cases.

2.3 SDN Controller

In the context of SDN, the SDN Controller represents the intelligence part in the SDN. Numerous studies have attempted to explain and compare between the different SDN network typologies based on using specific SDN controller. In Section 2.3.1 we sill investigate the difference in the control plane structures. In addition, in Section 2.3.2, we will highlight the difference between the SDN controller roles.

2.3.1 SDN Controllers Structures

As following, as shown in Figure 2.10 and as investigated in [36], the authors highlighted the difference in SDN Controllers Structures.

- **Physically-Centralized Control Plane**: A solo controller will be responsible for the whole SDN network. Despite the benefit of this in supporting the SDN simplicity principle, but this imposes two know weak points.
  Firstly, in this case the controller is a SPOF. Secondly, there will be a bottleneck for up-links to the SDN controller. Absloutly, this structure is not possible to be accepted or

Version	Major Feature	Reason	Use Cases
1.0 – 1.1	Multiple table	Avoid flow entry explosion	
	Group Table	Enable Applying action sets to group of flows	Load balancing, Failover, Link Aggregation
	Full VLAN and MPLS Support		
1.1-1.2	OXM Match	Extend matching flexibility	
	Multiple Controller	HA/Load balancing/Scalability	Controller Failover, Controller Load Balancing
1.2-1.3	Meter table	Add QoS and DiffServ capability	
	Table miss entry	Provide flexibility	
1.3-1.4	Synchronized Table	Enhance table scalability	Mac Learning/Forwarding
	Bundle	Enhance switch synchronization	Multiple switch configuration
1.4-1.5	Egress Table	Enabling processing to be done in output port	
	Scheduled bundle	Further enhance switch synchronization	

Table 2.9 – Features comparison for various OpenFlow versions [35]

Statistic	**OF1.0**	**OF1.1**	**OF1.2**	**OF 1.3&1.4**
Per table statistics	X	X	X	X
Per flow statistics	X	X	X	X
Per port statistics	X	X	X	X
Per queue statistics	X	X	X	X
Group statistics		X	X	X
Action bucket statistics		X	X	X
Per-flow meter				X
Per-flow meter band				X

Table 2.10 – Statistics support for various OpenFlow versions [34]

implemented in large networks as this design does not satisfy the various requirements of scalability.

- **Physically-Distributed Control Plane**: In contrast to the physically-centralized control plane structure, the Physically-Distributed Control Plane structure is using multiple-controllers to manage the topology. This considered as the solution to the probelms that exist in the Physically-Centralized structure.To take advantage of this model, more configurations for the east-west bound API are required.

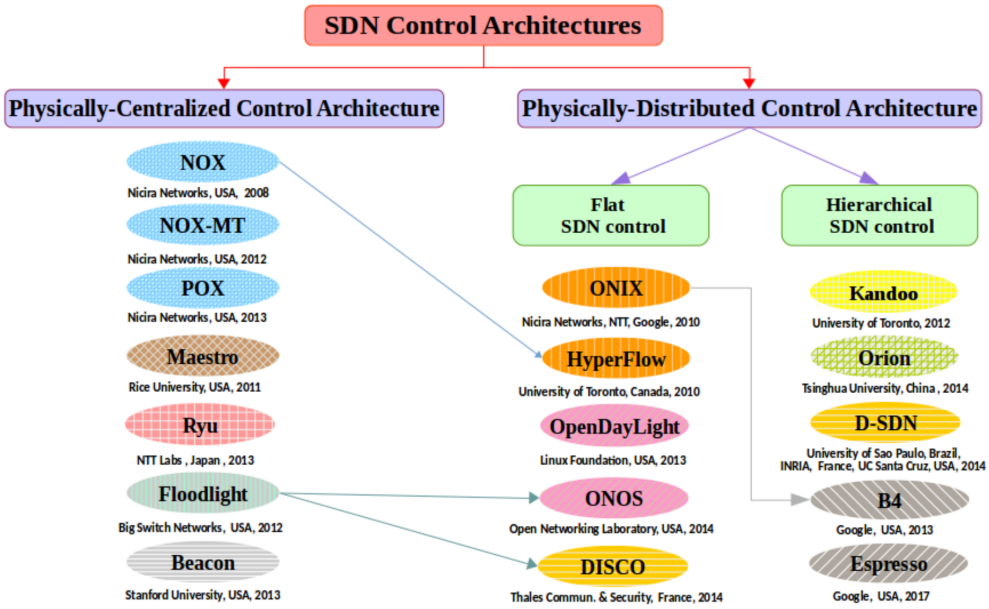


Figure 2.10 – SDN control plane architectures [36]

2.3.2 SDN Controllers Roles

As described In [26], there are three available SDN controller roles:

- **OFPCR_ROLE_EQUAL**: It is the default role of a SDN controller. In this role, the SDN controller has full read-write authorities. All other connected SDN controllers will be also have the same function.

- **OFPCR_ROLE_MASTER**: This role means that the SDN controller will act as the active SDN controller. Which means that it will receive all messages from the OpenFlow switches and write any commands to the OpenFlow switches.

- **OFPCR_ROLE_SLAVE**: In contrast to the Master role, SDN controller in this case will have only read access. Which means that the SDN controller will not get any switch

initiated messages like the asynchronous messages. The SDN controller is not allowed to send the controller-to-switch messages.

Chapter 3

Security Threats in SDN

3.1 Overview

As mentioned in Section 1.2, SDN offers many advantages and better network management performance. This is due to the separation of concerns between the control plane and the data plane. Although of all these benefits, previous studies have reported that SDN imposes many security challenges. In recent years, several SDN security studies have been published to highlight the lack of security in SDN. For example in [37] and based on the Confidentiality, Integrity and Availability (CIA) principle, the authors discussed the the attack surface and defenses in SDN. They classified the attacks to control plane remote attacks, control plane local attacks, control channel attacks and data plane attacks and under every category exist many vulnerabilities.
In [38], the author investigated the SDN southbound threats and identified the attack based on the source of the attack. According to [38], hosts , switches, controllers and the application plane can be a source of threats. Under each threat source exist several vulnerabilities that can be exploited by the attacker to affect the SDN network.
In [39], the author proposed eleven SDN attacks based on the the three planes structure. In the application plane 3 types of threats, in the control plane 3 other types of threats and finally in the data plane last 5 types of threats. The authors then mentioned 9 benefits of the SDN approach over the traditional network structure.
In [40], the authors provided an overall reference of the SDN vulnerabilities. They discussed the impact of the centralized SDN controller structure. Furthermore, the authors surveyed the security enhancements for SDN attacks. For further investigations regarding the impact of the centralized SDN controller structure, in [41], [42] the authors provided threat vectors and the potential consequences in SDN
.In [43], the authors investigated 7 attack types that exist in SDN and the impact on the CIA triad. In [44], the authors compared the security between the SDN and the traditional network

to know which one is more secure. They concluded that there is an urgent need for more SDN security mechanisms before any customer can accept it. In [**?**], they investigated the affect of an attack on the control plane and the impact of this on the SDN network. For more close look in a prominent SDN attack, in [45], [46] the authors investigated the impact of a DoS and DDoS attacks on SDN network. In [47], [48] more investigations for SDN vulnerabilities and proposed solutions.

3.2 Attacks on SDN

The total change in the network structure that was implemented by SDN imposes an urgent need for new security mechanisms. Different types of attacks targeting components, for example, the northbound API, the control plane, the southbound API, the data plane and the controller.As shown in Figure 3.1 and according to [6], in SDN, attacks have three main types and these attacks affect five components in SDN [6].

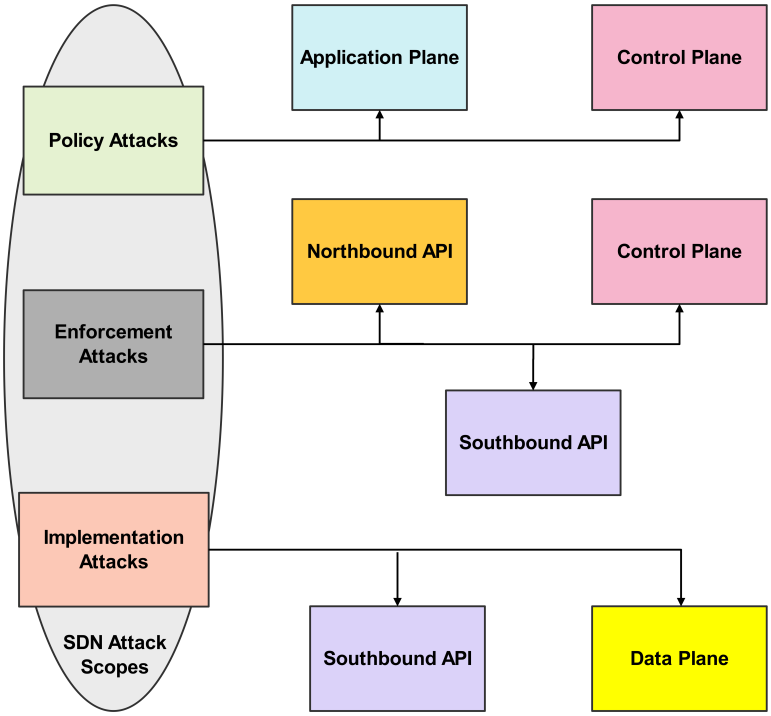


Figure 3.1 – SDN attacks scopes [6]

3.2.1 Implementation Attacks:

The first two components being a target for attacks are the data plane and the southbound API. As depicted in Figure 3.2, in implementation attacks [6], there are three types of attacks against the data plane and four types of attacks against the southbound API.

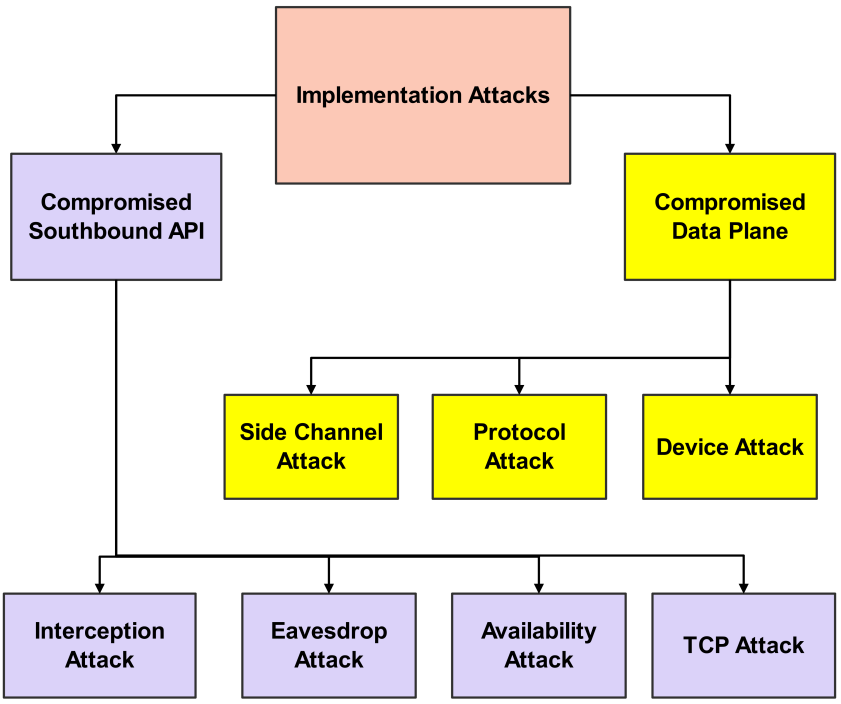


Figure 3.2 – SDN attacks scopes [6]

- **Device Attack**: A device attack is targeting the data plane by attacking the OpenFlow switch. This attack tries to profit from the existing vulnerabilities in both the software and the hardware. Existing hardware limitations like the limited Ternary Content-Addressable Memory (TCAM) space can exploited by the attacker [49], [50].This can leads to various attacks against the flow table in the OpenFlow switch like in [51]. Software flaws can also be exploited by the attacker. For example, lack of hosts authentications can be used to by the attacker to announce himself as another OpenFlow switch or even as a controller.
- **Protocol Attack**: Attackers can also exploit the vulnerabilities in the southbound API network protocols. Also the lack of support for all matching fields in many OpenFlow vendors switches leads to possible misuse from the attacker. In addition, DoS attacks are popular to be used to affect the data plane as discussed in [52], [53], [54].

- **Side Channel Attack**: Deduction of the forwarding policy by the attacker can be done through the packet processing times analysis or in general by analyzing the performance metrics.

In addition to these three attacks targeting the data plane, the following are four attack types against the southbound API:

- **Interception Attack**: By manipulating the messages being sent and received, attacker can demolish the network
- **Eavesdrop Attack**: Listening to know is the goal of the eavesdropping attack. The attacker will keep listening to the information being exchanged to get enough information to launch a more powerful attack.
- **Availability Attack**: This means simply the DoS and DDoS attacks. These attacks can lead to switch unavailability, so host isolation is the result. In general, DoS and DDoS can be done by flooding the southbound API by enormous number of inquiries. Simple tools like **MACOF** [55] can be used to interrupt the communication channel between the switch and the controller [38].
- **TCP Attack [37]**: The optional use of the Transport Layer Security (TLS) protocol enables the attacker to launch attacks targeting the channel between the controller and the OpenFlow switch.

3.2.2 Enforcement Attacks:

The main goal for the enforcement attacks is policies execution manipulation. This manipulation indicates that these types of attacks is affecting the time of policies execution, the place to implement these policies and by which way it can be implemented. As shown in Figure 3.3, there are three main attacks vectors exist in these types of attacks [6].

- **Southbound API**: By exploiting the previously mentioned in Section 2.2.1, the attacker can manipulate the policies. For example, the exploitation of the *OFPT_FLOW_MOD*, which is one of the controller-to-switch messages, the attacker can modify the flow table by adding, deleting, or modifying flow entries.
  Another important message is the *OFPT_PACKET_IN*, which is one of the asynchronous messages used to inform the controller about received packets that don't match any flow table entries.
  Added to the previous two messages types, disqualification of the controller role can

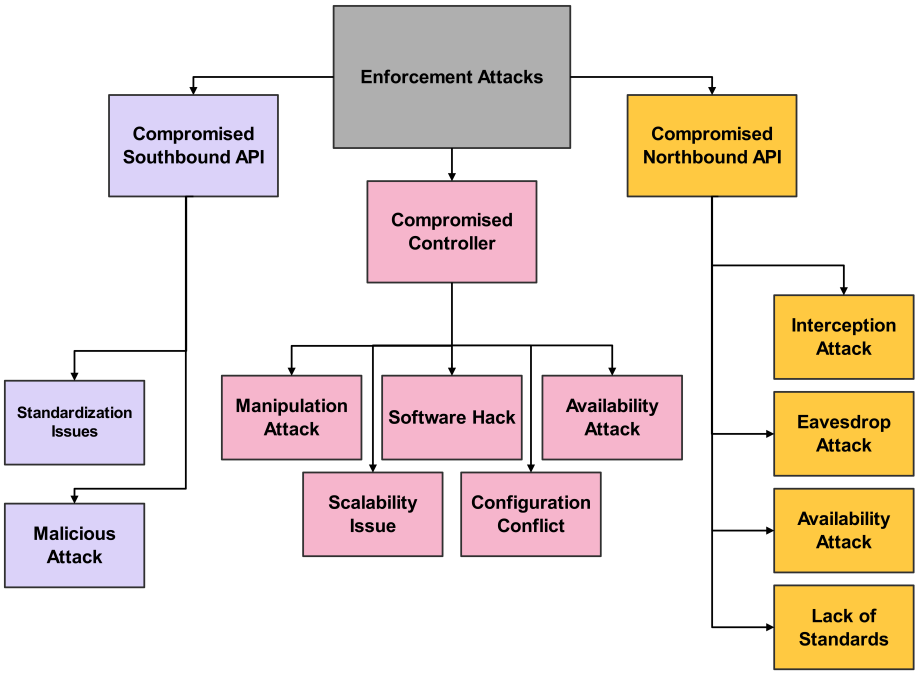


Figure 3.3 – Enforcement attacks in SDN [6]

be executed by the attacker by crafting the OFPT_ROLE_REQUEST message which is one of the controller-to-switch messages.
All previously mentioned messages can be used to launch the Man-In-The-Middle (MITM) attack which affect directly the southbound API [56].

- **Northbound API**: Another attack vector is targeting the northbound API. In case of bad design northbound API, attackers can exploit this to manipulate the network. Considering the significance of such important interface, it lacks the security features necessary to track and monitor the SDN applications [57]. In [57], the authors proposed a REST-like secure northbound API to tackle this problem. As mentioned in Section 2.1.4, the lack of standard secure northbound API can also be exploited.

- **Control plane**: Manipulations attack aims to trick the controller, so the result is wrong decisions. This manipulation depends on the misuse of the messages being transferred between the controller and the switch. Software hacks aim to miss-configure the server that hosts the SDN controller, so the result is a wrong functional controller or even out-of-service.

3.2.3 Policy Attacks:

As shown in Figure 3.4 [6], compromising the controller or the applications that manage the controller are the main goals of the policy attacks. Being totally dependant on third-party applications, the controller behavior and configuration can be altered by hacking those applications. Also, the attacker can exploit the absence of authentications and authorization between the controller and the applications. Besides, the absence of access control and accountability makes it easier for the attacker.

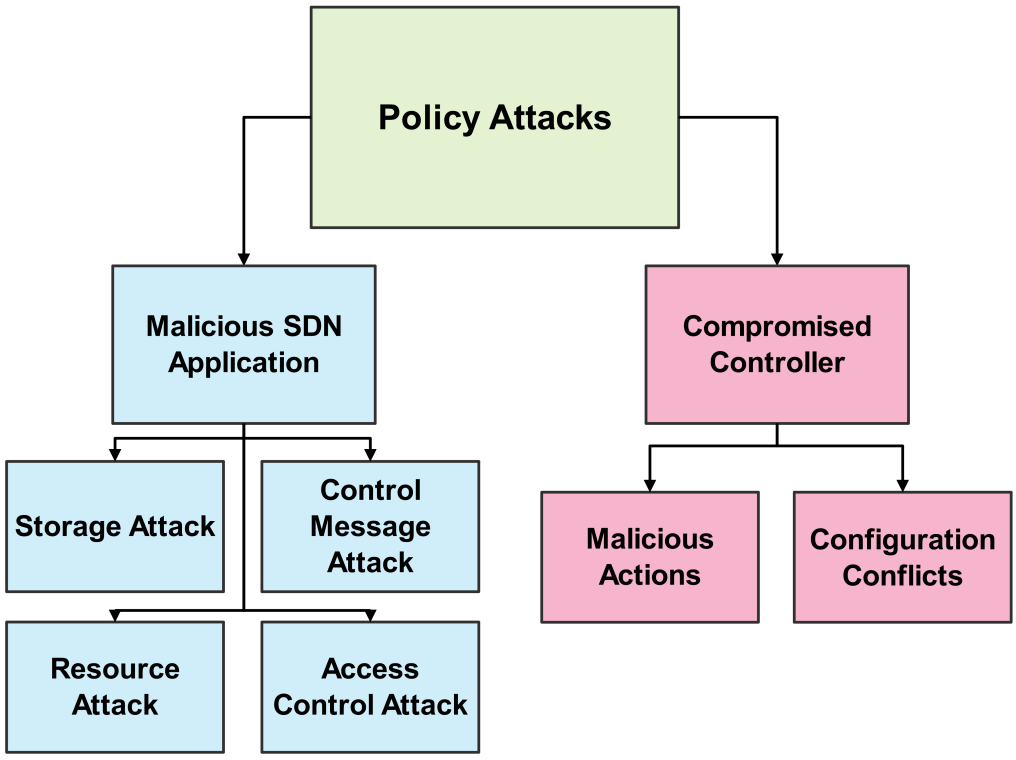


Figure 3.4 – Policy attacks in SDN [6]

3.3 DoS and DDoS Attacks:

Without any doubts, DoS and DDoS attacks are the most popular and have the biggest effect on services and systems [58]. DoS and DDoS attacks aim mainly to affect the availability of systems.
To easily imagine what DoS and DDoS attacks mean, let us assume that a large group of people intentionally called the hot-line of the medical service. This group starts telling long fake stories that they need medical assistance, which is not true. And you are an ordinary citizen having an accident, and you in dire need of an ambulance, and you called the Medical service hot-line to ask for urgent help. In this case, you will be on a long waiting list due to the calls from the group, as mentioned earlier. And the result will be the unavailability of medical assistance for you or, in best cases, the delay of it.
In DoS and DDoS attacks, the attackers flood the target will an enormous number of requests, and as a result, resource or bandwidth depletion occurred. Then the target will fail in providing the service for the legitimate user.
With the increased number of accessed network devices by many modern technologies like IoT, the risk of these devices being used as a source of attacks is increasing [58], [6].

3.3.1 DoS vs DDoS Attacks:

As illustrated in Figure 3.5, the difference between the DoS and DDoS attacks is based on the number of attack sources.
DoS attack is a system-on-system attack while DDoS attack is a multi-systems targeting one system [59].
The DDoS attacks use what is known as zombies. A **Zombie** is any compromised device that is controlled by the attacker and is used as a part of a botnet to launch an attack. A **Botnet** is a group of zombies used to launch DDoS attacks.

3.3.2 History and Statistics:

Back in 1974 was the first known DoS attack. In this attack a student at the age of 13 succeeded in launching a DoS attacks on PLATO terminals [61].
Back in 1996 was the first known DDoS attack. Unknown attacker launched a Synchronize (SYN) flood on Panix which is one of the Internet Service Providers (ISPs) in New York [62]. The downtime for the company services was 36 hours. Regarding the biggest known DDoS attacks, a DDoS attack of size 2.5 Tbps was carried out against Google in 2017 [63]. In Q1 2020, another massive DDoS attack of size 2.3 Tbps was carried out against Amazon Web Services (AWS) in 2020 [64]. Further information about early known massive DDoS attacks can be seen in Figure 3.6

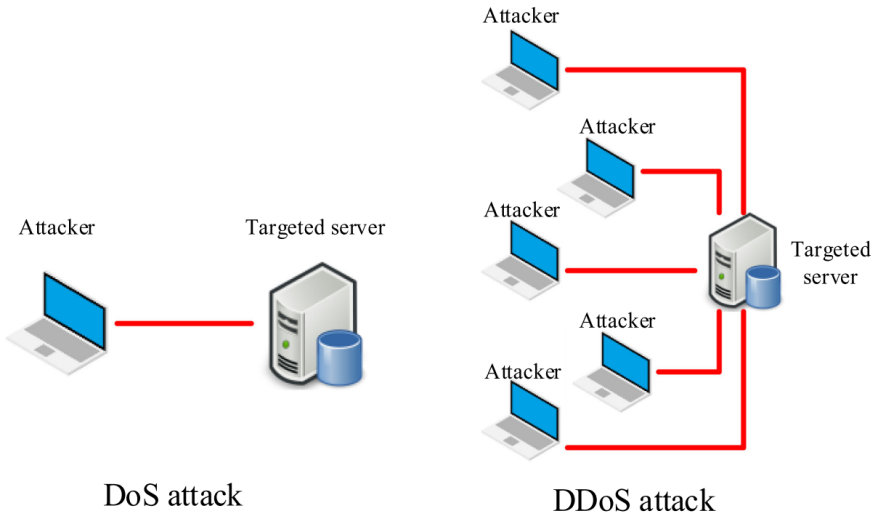


Figure 3.5 – DoS attacks vs DDoS attacks [60]

3.3.3 DDoS Attack Classifications:

Based on the criteria used, the classification of the DDoS attacks can be different. The following are some classifications methods.

- **Open Systems Interconnection (OSI) Layer-dependant DDoS Attacks [65]:** DDoS attacks can be classified based on the targeted OSI layer.
  - **Application Layer Attack:** Using protocols like Hypertext Transfer Protocol (HTTP) Hypertext Transfer Protocol Secure (HTTPS), with the help of the Create, Read, Update and Delete (CRUD) operations, the attacker can attack the top layer in the OSI model. The application layer is where applications reside and attacking this layer affects the functionality of the applications, which will result in user bad experience.
  - **Network Transport Layers Attacks:** This attack focuses in targeting the links or the memory of the network devices. Taking the advantages of some protocols vulnerabilities, attacker can launch DDoS attacks. Famous used protocols to launch an attack on layers 3 and 4 are TCP and UDP.
- **Direct and Reflector-Based DDoS Attacks [65]:** Based on the type of the used machines to launch the DDoS Attacks, the DDoS Attacks can be classified into the following:
  - **Direct Attacks:** Attackers can use legitimate victims known as zombies to launch DDoS Attacks directly. Using zombies has two benefits for the attacker: first, identity hide; second, attack volume increases.

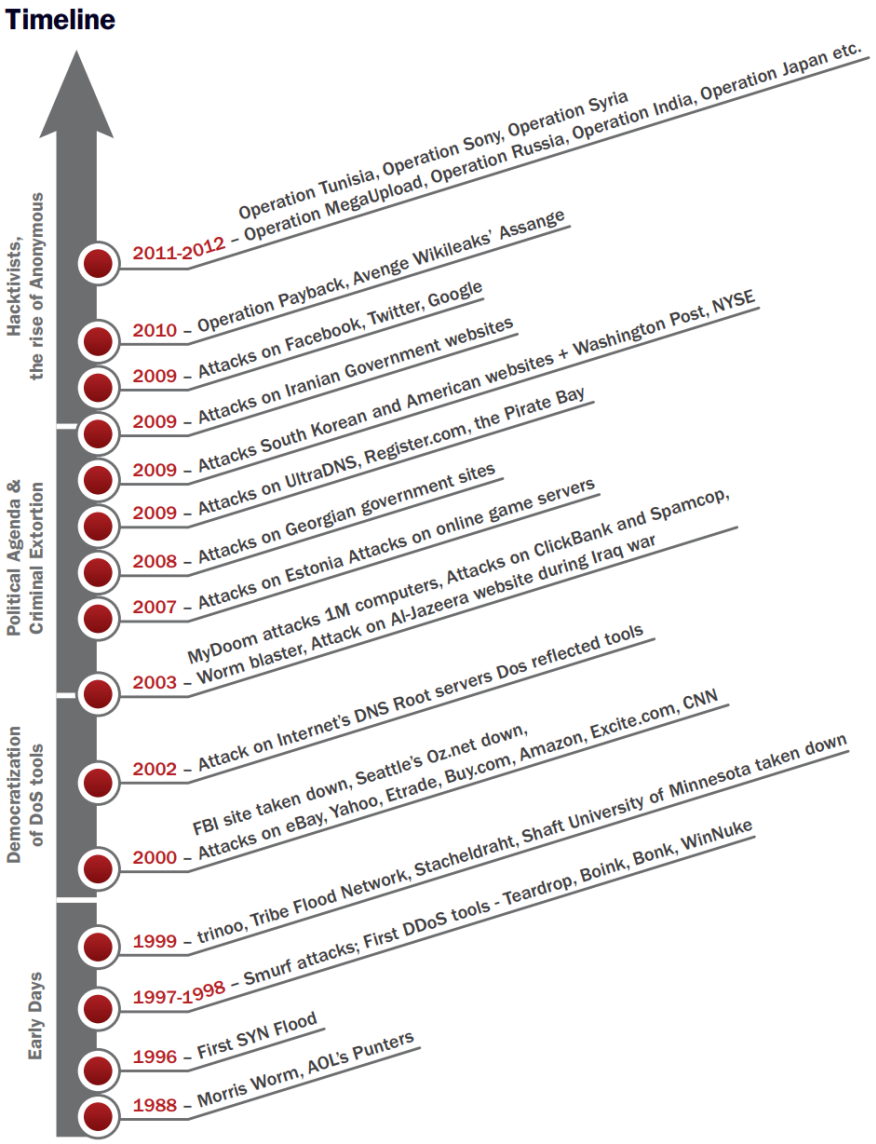


Figure 3.6 – Famous known DDoS attacks [61]

  - **Reflector Attacks:** Attackers can spoof the source the Internet Protocol (IP) address of the victim and send requests to what is called **reflectors** which are intermediate innocent machines, and as a sequence, those servers will respond, and these answers will form a DDoS attack.

- **Direct and Indirect DDoS Attacks [65]:** Based on directly attacking the victims or indirectly, the DDoS Attacks can be classified into the following:
  - **Direct Attacks:** These attacks aim to attack the victim directly. In these attacks, the attacker floods the physical machine with massive requests.
  - **Indirect Attacks:** The attackers will target the victim indirectly. This attack aims to target the highly paramount services to the victim, which leads to the victim machine performance degradation.

3.3.4 Famous DDoS Attacks:

The Q4 2020 report in Figure 3.7 sketches the famous DDoS attacks. In the following section different famous DDoS attacks are explained.

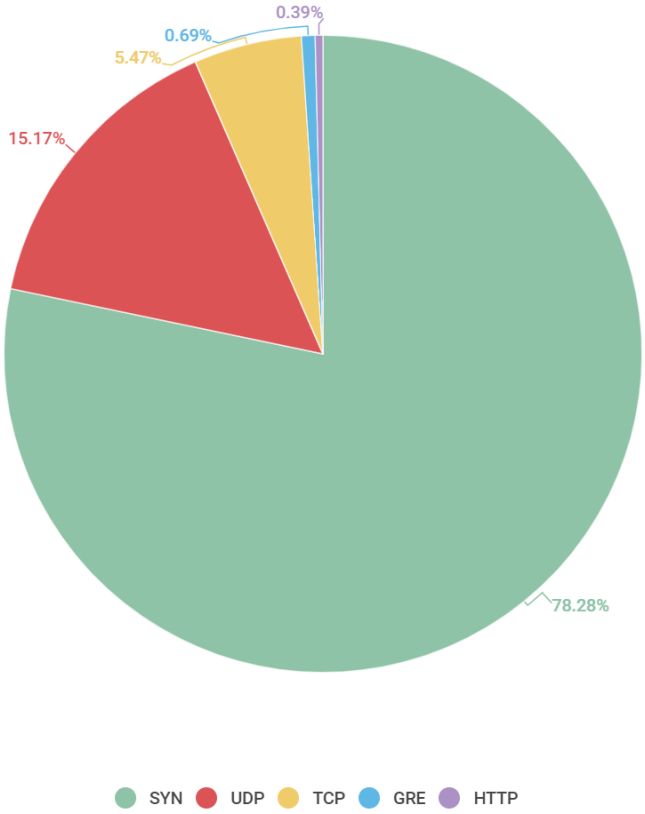


Figure 3.7 – The famous known DDoS attacks [66]

- **TCP SYN Attacks [67]:**
  In TCP SYN attacks, the attacker is exploiting the three-way handshake process in the TCP protocol. The attacker is sending a large number of SYN requests to the server. Then, the server will respond with an SYN-ACK message, open a communication port, and waits for a response from the source. But the attacker will not send the final Acknowledgement (ACK) message. As a result, the server has incomplete connections and cannot serve legitimate users due to network resources depletion. This attack is also known as a half-open attack. Figure 3.8 highlights the structure of launching a TCP SYN attack.

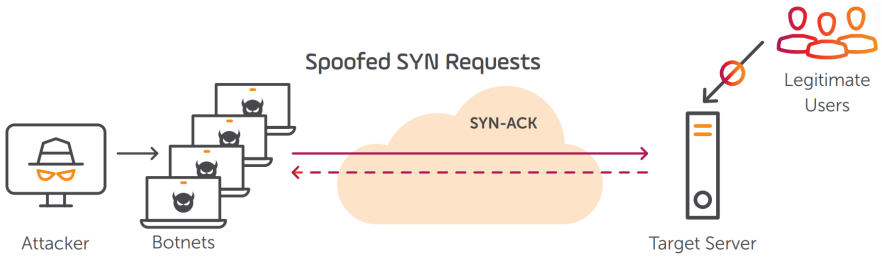


Figure 3.8 – TCP SYN Attack [67]

- **UDP Attacks [67]:**
  UDP protocols is a connection-less protocol, which means, user, can initiate a connection to a server only by sending an UDP packet. The attacker can affect server availability only by sending an enormous number of UDP packets. As a result, the server will fail to serve legitimate users due to the available port limit number. Figure 3.9 highlights the structure of launching UDP attacks.

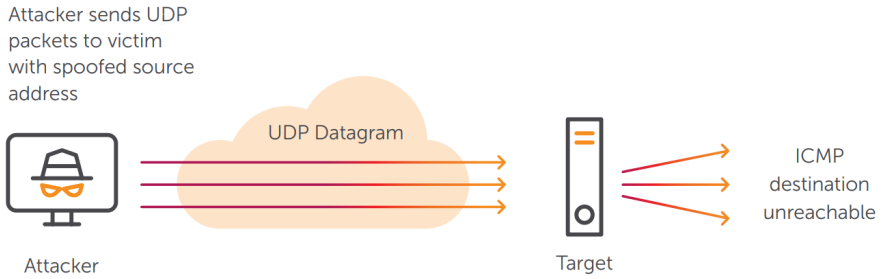


Figure 3.9 – UDP Attacks [67]

- **Domain Name Server (DNS) Flood Attacks [67]:**
  The attacker can flood the victim indirectly by massive DNS response by sending a sizeable high-speed number of spoofed DNS queries impersonating to be as if they been sent from the victim side.
  Figure 3.10 highlights the structure of launching DNS attack.

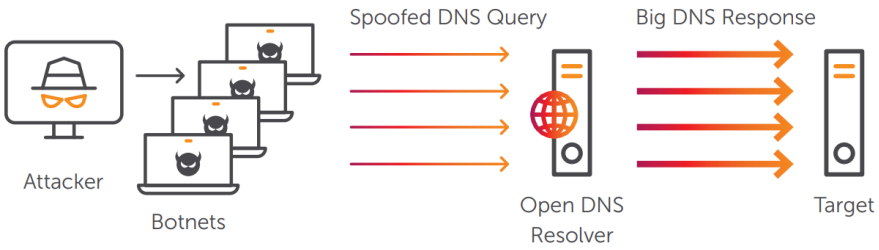


Figure 3.10 – DNS Flood Attack [67]

3.3.5 Famous DDoS Datasets:

To explain the impact of a using DDoS attacks on SDN, an overview of the recent popular DDoS datasets is needed. The following is explanation for four DDoS traffic datasets:

3.3.5.1 NDSec-1 on AWS

The NDSec-1 dataset [68] is created in the Network and Data Security Group's facilities at the University of Applied Sciences in Fulda, Germany, and contains traces and log files of cyber-attacks. The lack of publicly accessible captures containing a wide range of different attack signatures to either test existing anomaly detection methods or help network security studies in developing new detection engines prompted the development of such a dataset. The dataset offers four classes of cybersecurity attacks (BYOD, Watering hole, Botnet, and others), and each class contains three files.
The first file is the Packet Capture Next Generation (PCAPNG) file containing the network traffic. The second file is an excel file in Comma-Separated Values (CSV) format containing all log events. The third and last file is the ground truth file that is also in the CSV format that includes the annotation/comments about benign or malicious traffic.
The botnet class is what is related to DDoS attacks detection assessment. The PCAPNG file contains 135555 packets and the ground truth file interprets the content of these packets. The ground truth file has various DDoS attacks, for example, SYN-Flood. Besides, it contains Command and Control (C&C) communications traces as the compromised victim is a part of a botnet and needs a connection to the attacker's server which is known as the bot master. For further analysis of this dataset, this is discussed in [69], [70].

3.3.5.2 CICIDS2017

The Canadian Institute for Cybersecurity (CIC) released the CICIDS2017 dataset [71] in 2017. CICIDS2017 dataset contains both benign and attacks traffic in both Packet Capture (PCAP) and CSV formats. The PCAP file is the real data captured, which helps replay the traffic if needed. The CSV file contains the traffic along with more than 80 extracted features, and the

file is generated using the CICFlowMeter [72] tool.

The traffic was captured in 5 days, and every day contains a different type of traffic. For more information about the content of each day's traffic, this can be seen in Figure 3.11. It is worth mentioning that CICIDS2017 has a highly imbalanced dataset because the total percentage of the benign traffic is 80.3% of the total number and the unequal distribution of the classes.

File Name	Type of Traffic	Number of Record
Monday-WorkingHours.pcap_ISCX.csv	Benign	529,918
Tuesday-WorkingHours.pcap_ISCX.csv	Benign	432,074
	SSH-Patator	5,897
	FTP-Patator	7,938
Wednesday-WorkingHours.pcap_ISCX.csv	Benign	440,031
	DoS Hulk	231,073
	DoS GoldenEye	10,293
	DoS Slowloris	5,796
	DoS Slowhttptest	5,499
	Heartbleed	11
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Benign	168,186
	Web Attack-Brute Force	1,507
	Web Attack-Sql Injection	21
	Web Attack-XSS	652
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Benign	288,566
	Infiltration	36
Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign	189,067
	Bot	1,966
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Benign	127,537
	Portscan	158,930
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Benign	97,718
	DdoS	128,027
Total Instance/ Record		**2,830,743**

Figure 3.11 – CICIDS2017 dataset summary [73]

3.3.5.3 CSE-CIC-IDS2018 on AWS

CSE-CIC-IDS2018 [74] was released in cooperation between CIC and Communications Security Establishment (CSE) in 2018. CSE-CIC-IDS2018 dataset contains traffic captured in 10 days with 83% benign traffic percentage. As depicted in Figure 3.12, the dataset containsDDoS traffic generated by the Low Orbit Ion Canon (LOIC) tool. In Figure 3.13, the network traffic distribution of the CSE-CIC-IDS2018 dataset is shown.

Attack	Tools	Duration	Attacker	Victim
Bruteforce attack	FTP – Patator SSH – Patator	One day	Kali linux	Ubuntu 16.4 (Web Server)
DoS attack	Hulk, GoldenEye, Slowloris, Slowhttptest	One day	Kali linux	Ubuntu 16.4 (Apache)
DoS attack	Heartleech	One day	Kali linux	Ubuntu 12.04 (Open SSL)
Web attack	• Damn Vulnerable Web App (DVWA) • In-house selenium framework (XSS and Brute-force)	Two days	Kali linux	Ubuntu 16.4 (Web Server)
Infiltration attack	• First level: Dropbox download in a windows machine • Second Level: Nmap and portscan	Two days	Kali linux	Windows Vista and Macintosh
Botnet attack	• Ares (developed by Python): remote shell, file upload/download, capturing • screenshots and key logging	One day	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)
DDoS+PortScan	Low Orbit Ion Canon (LOIC) for UDP, TCP, or HTTP requests	Two days	Kali linux	Windows Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit)

Figure 3.12 – CSE-CIC-IDS2018 dataset summary [74]

Traffic type	Distribution (%)
Benign	83.070
DDoS	7.786
DoS	4.031
Brute force	2.347
Botnet	1.763
Infiltration	0.997
Web attack	0.006

Figure 3.13 – CSE-CIC-IDS2018 network traffic distribution [75]

3.3.5.4 CICDDoS2019

CICDDoS2019 [76] [77] [78] is the biggest modern DDoS dataset that contains recent DDoS attacks and benign traffic. The dataset is available in both CSV and PCAP formats. The PCAP file is the actual real-world captured data, which helps replay the traffic if needed. The CSV file contains the traffic along with more than 80 extracted features, and the file is generated using the CICFlowMeter [72] tool.
In comparison to all previously mentioned DDoS datasets, CICDDoS2019 is more focused towards only DDoS attacks. In addition, it is modern, high quality and contains wide DDoS attacks types. The dataset is generated using real machines as listed in Table 3.1 to simulate the real attacks scenarios. To simulate the benign traffic, traffic for 25 users based on HTTP, HTTPS, File Transfer Protocol (FTP), Secure Shell (SSH), and email protocols was generated. For full list of available DDoS attacks in CICDDoS2019, see Figure 3.14, Table 3.2. As mentioned above, The 18 CSV files contains more than 80 extracted features from the 964 PCAP files. For full list of the features and their description see Table 3.3.

Machine	OS	IPs
Server	Ubuntu 16.04 (Web Server)	192.168.50.1 (first day)
		192.168.50.4 (second day)
Firewall	Fortinet	205.174.165.81
PCs (first day)	Win 7	192.168.50.8
	Win Vista	192.168.50.5
	Win 8.1	192.168.50.6
	Win 10	192.168.50.7
PCs (second day)	Win 7	192.168.50.9
	Win Vista	192.168.50.6
	Win 8.1	192.168.50.7
	Win 10	192.168.50.8

Table 3.1 – Operation Systems and IPs of the Victims Machines

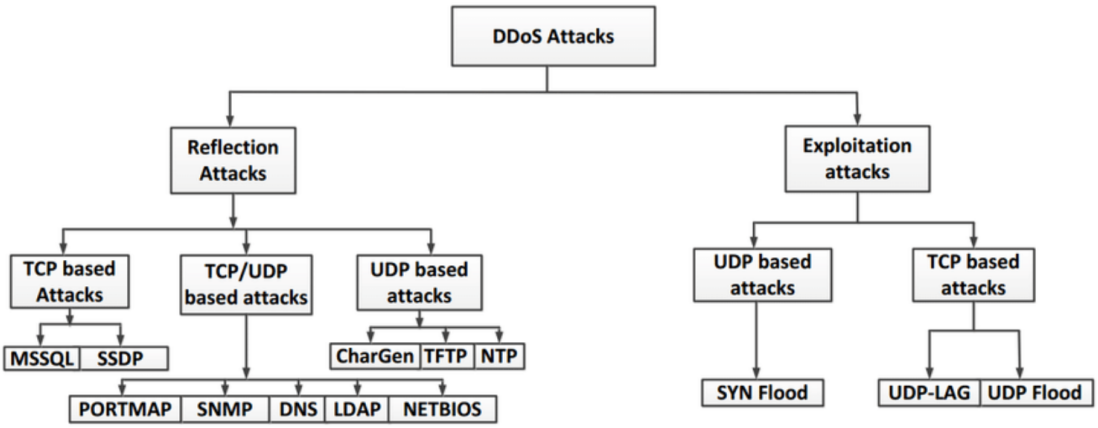


Figure 3.14 – CSE-CIC-IDS2018 network traffic distribution [77]

Days	Attacks	Attack times
Training Set (First Day 12.01)	PortMap	09 : 43 – 09 : 51
	NetBIOS	10 : 00 – 10 : 09
	LDAP	10 : 21 – 10 : 30
	MSSQL	10 : 33 – 10 : 42
	UDP	10 : 53 – 11 : 03
	UDP-Lag	11 : 14 – 11 : 24
	SYN	11 : 28 – 17 : 35
	NTP	10 : 35 – 10 : 45
Testing Set (Second Day 11.03)	DNS	10 : 52 – 11 : 05
	LDAP	11 : 22 – 11 : 32
	MSSQL	11 : 36 – 11 : 45
	NetBIOS	11 : 50 – 12 : 00
	SNMP	12 : 12 – 12 : 23
	SSDP	12 : 27 – 12 : 37
	UDP	12 : 45 – 13 : 09
	UDP-Lag	13 : 11 – 13 : 15
	WebDDoS (ARME)	13 : 18 – 13 : 29
	SYN	13 : 29 – 13 : 34
	TFTP	13 : 35 – 17 : 15

Table 3.2 – CICDDoS2019 days attacks and attacks times

Feature Name	Description
	In the forward direction
Total Fwd Packets	The total # of packets sent
Fwd IAT Min	The minimum time between two packets
Fwd IAT Max	The maximum time between two packets
Fwd IAT Mean	The mean time between two packets
Fwd IAT Std	The standard deviation time between two packets
Fwd IAT Total	The total time between two packets
Fwd PSH flags	# of times the PSH flag was set in packets
Fwd URG Flags	# of times the URG flag was set in packets
Fwd Header Length	The total bytes used for headers
Total Length of Fwd Packet	The total size of the packet
Fwd Packet Length Min	The minimum size of packet
Fwd Packet Length Max	The maximum size of packet
Fwd Packet Length Mean	The Mean size of packet
Fwd Packet Length Std	The standard deviation size of packet
FWD Packets/s	# of forward packets per second
Fwd Segment Size Avg	Average size observed in the forward direction
Fwd Bytes/Bulk Avg	Average number of bytes bulk rate
Fwd Packet/Bulk Avg	Average number of packets bulk rate
Fwd Bulk Rate Avg	Average number of bulk rate
Subflow Fwd Packets	The average number of packets in a sub flow
Subflow Fwd Bytes	The average number of bytes in a sub flow
Fwd Init Win bytes	The total number of bytes sent in initial window
Fwd Act Data Pkts	Count of packets with at least 1 byte of TCP data payload
Fwd Seg Size Min	Minimum segment size observed
	In the backward direction
Total Bwd Packets	The total # of packets
Bwd IAT Min	The minimum time between two packets
Bwd IAT Max	The maximum time between two packets
Bwd IAT Mean	The mean time between two packets
Bwd IAT Std	The standard deviation time between two packets
Bwd IAT Total	The total time between two packets
Bwd PSH Flags	# of times the PSH flag was set in packets
Bwd URG Flags	# of times the URG flag was set in packets
Bwd Header Length	The total bytes used for headers
Total Length of Bwd Packet	The total size of the packet
Bwd Packets/s	# of backward packets per second
Bwd Packet Length Min	The minimum size of packet
Bwd Packet Length Max	The maximum size of packet
Bwd Packet Length Mean	The mean size of packet
Bwd Packet Length Std	The standard deviation size of packet

Bwd Segment Size Avg	Average number of bytes bulk rate in the backward direction
Bwd Bytes/Bulk Avg	Average number of bytes bulk rate
Bwd Packet/Bulk Avg	Average number of packets bulk rate
Bwd Bulk Rate Avg	Average number of bulk rate
Subflow Bwd Packets	The average number of packets in a sub flow
Subflow Bwd Bytes	The average number of bytes in a sub flow
Bwd Init Win bytes	The total number of bytes sent in initial window
	Direction independent
Flow duration	The flow's duration measured in microseconds
Flow Bytes/s	# of flow bytes per second
Flow Packets/s	# of flow packets per second
Flow IAT Mean	The Mean time between two packets
Flow IAT Std	The standard deviation time between two packets
Flow IAT Max	The maximum time between two packets
Flow IAT Min	The minimum time between two packets
Packet Length Min	Maximum length of a packet
Packet Length Mean	Mean length of a packet
Packet Length Std	Standard deviation length of a packet
Packet Length Variance	Variance length of a packet
FIN Flag Count	# of packets with FIN Flag
SYN Flag Count	# of packets with SYN Flag
RST Flag Count	# of packets with RST Flag
PSH Flag Count	# of packets with PUSH Flag
ACK Flag Count	# of packets with ACK Flag
URG Flag Count	# of packets with URG Flag
CWR Flag Count	# of packets with CWR Flag
ECE Flag Count	# of packets with ECE Flag
Down/Up Ratio	Download and upload ratio
Average Packet Size	The packet average size
Active Min	Minimum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Idle Min	Minimum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

Table 3.3 – CICDDoS2019 Features and their description [72]

Chapter 4

Related Work

In this chapter an overview about the related work to the investigation of the detection and mitigation of DDoS attacks on SDN is expained.

4.1 Statistical Methods

As discussed in chapter 2, SDN is prone to various attacks based on the target of the attacker. The effect of a Distributed Denial of Service (DDoS) attack is one of these crucial challenges. Entropy is a statistical method and, as a measure of uncertainty, has been used in many types of research to detect DDoS attacks.
In [79] and using the POX controller [80], the authors proposed an effective and lightweight solution for DDoS attack detection in SDN. The solution is analyzing the first 250 packets as window size and extracting the destination IP address value. Based on this analysis and a predefined threshold for the entropy value, the model can determine if this traffic is DDoS attack or not.
In [81] and using the same idea, the authors applied it on different test cases using a variable number of the POX controllers in a multiple controllers topology. The same idea of using entropy to detect the DDoS is used in [82] and [83].
In [84] and [85], the authors proposed a detection method based on the ϕ-Entropy. The benefit of the ϕ-Entropy is that it amplifies the characteristics variations between different data types. In [86], the authors proposed mitigation for the DDoS attacks by blocking the port of the attacker.

4.2 Deep Learning Methods

Although all the advantages of the statistical methods, it suffers from many lacks.
In [87], the authors listed many of those lacks. The most pertinent point is the need for

experience to tune and adjust the threshold and the window size values based on the different scenarios and the limit of using the statistical features for only specific DDoS attack vectors. Another interesting point to consider, the SDN infrastructure is prone to the slow rate DDoS attacks in case of using the statistical methods. Another pertinent point, as discussed in [88], the statistical methods fail in distinguishing between the DDoS attacks and flash crowds traffic. The most difficult challenge will be the low accuracy values of the statistical methods for optimal, accurate scenario detection. Finally, the accuracy is not enough as a sole precise metric.

The trend of using machine learning and deep learning algorithms in detecting the DDoS attacks is increasing.

In [89], The authors used the Support Vector Machine (SVM) model, a supervised machine learning model, to classify the traffic into two groups Benign or DDoS attack. The authors used six-tuple features to be collected from the traffic during the training phase. For DDoS attacks, the authors used Hping3 [90] tool to generate Internet Control Message Protocol (ICMP), UDP and TCP traffic.

In [91], the authors proposed a new model to detect DDoS attacks on SDN.For a dataset, NSL-KDD is selected with a total of 25 selected features. NSL-KDD [92] is updated version of KDD cup99 [93] dataset and contains 4 classes of attacks (DoS, Probe, User to Root (U2R) and Remote to Local (R2L)). The authors used three selection algorithms for feature selection (Genetic, Ranker, Greedy). In addition, the authors used the Naive Bayes (NB) classifier to classify the traffic. NB classifier is a probability-based machine learning model that uses the Bayes' theorem to discriminate the difference between various objects [94].

In [95], the authors combined the SVM and K-Nearest Neighbors (KNN) algorithms to detect the DDoS attacks on SDN. The authors used trafgent [96] tool to generate the traffic and collected five-tuple features. The KNN is a supervised machine learning algorithm that can be used to solve problems involving both classification, and regression [97]. KNN considers that identical items are close to each other and are using for example, the Euclidean distance to calculate the distance between neighbors.

Although machine learning algorithms achieved numerous improvements in DDoS attacks detections, it suffers from specific lacks compared to deep learning algorithms.

Firstly, machine learning algorithms need more human intervention for feature extraction than deep learning algorithms. **Secondly**, machine learning algorithms suffer from flow collection bottleneck due to the need for feature extraction [88] . **Lastly**, the resultant achieved accuracy is not high enough to guarantee the validity of using the machine learning algorithms in real-time environments [88]. Given the preceding, the need to dive deep and do more researches in the deep learning area is inevitable.

In [98], the authors proposed Safe-Guard Scheme (SGS) based on two modules. The first

module is anomaly traffic detection used in the data plane, and the second module is the dynamic controller defense used in the control plane.
The anomaly traffic detection module adopts Byte Rate (BR), Symmetric Flows Percentage (SFP), Variation Rate of Asymmetric Flow (VAFR) and Flows Percentage with Small Amount Packets (FPSA) as a four-tuple vector. It focuses on the switches in the data plane to distinguish the benign traffic from the attack/malicious traffic. The four-tuple features will be the input to a Back Propagation Neural Network (BPNN) network, which is trained before using a trained dataset. In the controller dynamic defense module, the focus is to remap the targeted SDN controller to other SDN controller and guide the switches about the change of roles. The limitations of this paper is feeding the BPNN network with values not images.
In [99], the authors proposed a deep ensemble Convolutional Neural Network (CNN) framework to detect DDoS attacks on SDN. The authors used a ten layers CNN framework, including four 2D-Conv layers, two MaxPooling 2D layers, two Fully-Connected (FC) layers, one flatten, and one sigmoid classifier layer. The model achieves better results in accuracy, precision, recall, and F1 score based on the comparison with state-of-the-art research. The performance of the CNN is improved if the dataset to train the model is images [100].
In [100], the authors proposed a novel Intrusion Detection System (IDS) to detect intrusions using Fast Fourier Transform (FFT). The idea was to apply FFT the sampling to the traffic. The idea was to obtain 4096 sampling points and then convert them to 64*64 2D Images. The challenge was that each image has three channels Red Green Blue (RGB), so the authors considered the real part values as the first channel, the imaginary part values as the second channel, and the summary between both as the third channel.
In [88], the authors presented the idea of converting the captured traffic packets to image by converting each byte, which equals 8 bits, equal to two hexadecimal values representing a pixel in the image. The limitation of this idea is using the same value for the three channels which always results in a gray-scale image.
Marwan *et al.* in [101] used a Recurrent Neural Network (RNN) model to detect intrusions in SDN.The author used three datasets (KDD Cup 1999, NSLKDD, and UNSW-NB15) to test the proposed approach. In addition, the author clarifies the usage of three sections architecture to detect and mitigate anomalies (flow collector, anomaly detector, and anomaly mitigator).
Mahmoud *et al.* in [102], proposed a model based on RNN and autoencoder algorithms. The author used [76] dataset to evaluate the proposed approach and proves that the proposed approach outperforms the shallow learning methods.

Chapter 5

Simulation Setup

This chapter contains information about the hardware and software tools used to create the SDN topology and simulate the DDoS attacks behaviour in SDN.

- **Hardware**: for hardware options, many options exist to simulate the required topology:
  - **Cloud**: There are currently many options to host your machine in the cloud and access it from anywhere in the world. For example, AWS, Google Cloud Platform (GCP) and Microsoft Azure. Another option is to use online platforms like kaggle or Google Colab to run the code.
  - **On Premise**: Using either your local PC or all-in-one pre-built virtual machine or all-in-one pre-built docker instance.

  For the used hardware to simulate the required topology, the specifications can be seen in the table Table 5.1:

System Manufacturer	Google Cloud Platform (GCP)
Processor	AMD EPYC 7B12 (16 vCPU)
Memory	32GB
Operating System	Ubuntu 20.04
Hard Disk	120 SSD

Table 5.1 – Hardware Specifications

- **Software**:For a list of the used software programs, Table 5.2 contains all used software programs with their version.

we use Mininet [103] Ryu Controller [104] further described in detail in Section 5.1 and Section 5.2.

Program	Purpose	version
CIC-FlowMeter	Convert PCAP files to CSV features-included files	3
Free Huge CSV Splitter	Split the CSV files	-
Mininet	Run the SDN topology	2.3.0
RYU	Manage the SDN topology	4.34
Vandyke SecureCRT	Access the cloud-hosted Ubuntu	8.5.4
Visual studio code	Create and Edit Python codes	1.55.2
WEKA	Analysis & visualize the dataset	3.8.5
Wireshark	Analysis & capture the traffic	3.4.4
WinSCP	Transfer data between Local PC and the cloud	5.17.10
EdrawMax	Draw network diagrams	10.5.0

Table 5.2 – Software Specifications

5.1 Mininet

Mininet is a Linux-based network emulator that allows you to build network diagrams, hosts, switches, and controllers in a virtual machine. It supports OpenFlow protocol and provides both Command-Line Interface (CLI) and Graphical User Interface (GUI) (**Miniedit**) capabilities to create and configure network topologies.
Mininet offers many advantages as following [103]:

- Easy-to-learn and inexpensive network testbed.
- Easy Installation, either with simple commands or using pre-build virtual machine.
- Provides a Python API.
- Support Integration with the real environments.
- Easy integration with famous SDN controllers.
- Support Inter-controllers communications between different controllers [105].

5.2 Ryu Controller

Ryu Controller [106] is python-based open-source SDN controller. It supports various versions of the OpenFlow protocol, like 1.0,1.2,1.3,1.4. In addition, it supports many other protocols like SNMP, Virtual Router Redundancy Protocol (VRRP) Netconf.
In Figure 5.1 shows the architecture of the Ryu controller. It contains four status that enables the creation and troubleshooting of SDN topologies. The four status are the following:

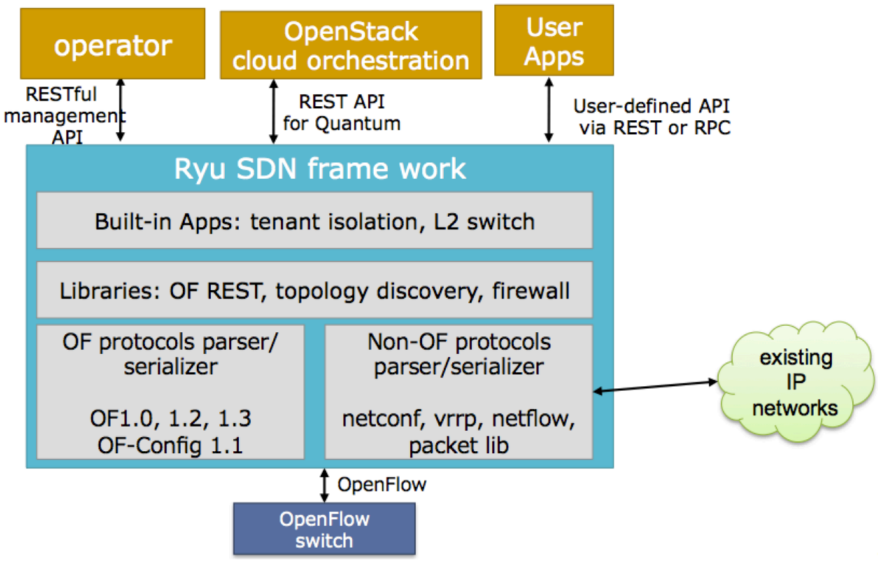


Figure 5.1 – Ryu architecture [107]

- **HANDSHAKE_DISPATCHER**: The controller and the switch are sending each other hello messages to recognize each other.
- **CONFIG_DISPATCHER**: The initial setup started, and the controller will ask for the switch features.
- **MAIN_DISPATCHER**: This is the running phase in which the controller starts to receive/send packets in/out from/to the switch.
- **DEAD_DISPATCHER**: The tear-down of the connection between the controller and the switch.

As shown in Figure 5.2, Ryu supports Restful API through using the *ofctl_rest.py* module so the network manager can interact with it easily through simple commands.

The relation between the event queue, the event loop thread and the event handler is shown in Figure 5.3.

Furthermore, the Ryu controller provides a GUI capability to see the resulting network through using the topology viewer. In addition, it offers the integration with the Sampled FLOW (sFLOW) tool to show useful graphs about the topology.

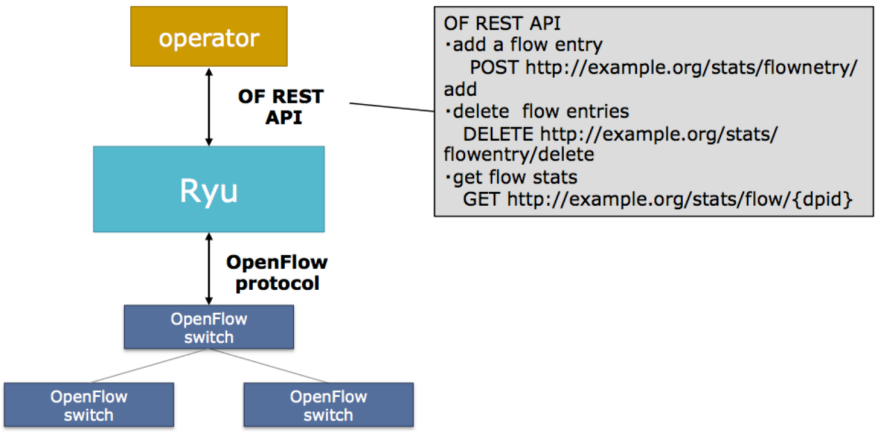


Figure 5.2 – Ryu controller Restful [107]

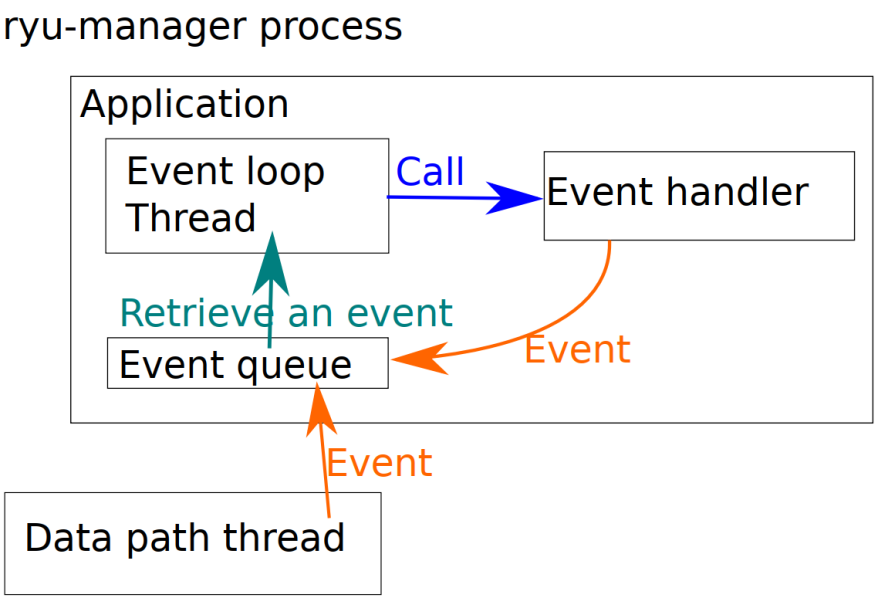


Figure 5.3 – Ryu programming model architecture [106]

Chapter 6

Proposed Approach

As explained in Chapter 4, further researches investigated the validity of using either the machine learning or the deep learning algorithms in the detection of DDoS attacks on SDN. This chapter aims to explain and clarify the proposed approach of converting the chosen dataset to images then use the VGG model to detect and classify DDoS attacks on SDN. Based on the output result of the VGG16 model, the controller can process or drop the flow.

6.1 Overview

The proposed approached is using VGG16 very deep learning model and for evaluation CICDDoS2019 dataset is used. The chosen state-of-the-art approaches to compare with are [88] and [77]. For details about [88] and [77] researches see Table 6.1 and Table 6.2 respectively:

Detection algorithms	CNN
Structure	Four modules (Rate detection, Entropy detection, Data processing & Deep learning detection)
DDoS attacks	HTTP flooding, UDP flooding, ICMP flooding & SYN flooding
Dataset	CICIDS2017
Environment	Mininet+POX
Traffic generation tool	Hping3
Evaluation Metrics	Accuracy, Precision, Recall, F1-Score & Training time.

Table 6.1 – Overview about the research discussed in [88]

Detection algorithms	Iterative Dichotomiser 3 (ID3), Random Forest (RF), Naïve Bayes, and Linear Regression (LR)
Structure	Two separated networks (Victim-Network & Attack-Network) including all real used devices like switches, routers, firewall (Fortinet) and PCs (Ubuntu, Win vista, Win 7, Win 8.1 and Win 10)
DDoS attacks	All attacks mentioned in Table 3.2
Dataset	CICDDoS2019
Traffic generation tool	PCs with two profiles system (B-Profile and M-Profile)
Evaluation Metrics	Precision, Recall & F1-Score

Table 6.2 – Overview about the research discussed in [77]

6.2 Model steps

6.2.1 CSV files

As mentioned before in Section 6.1, the CICDDoS2019 dataset will be used to test and evaluate the proposed approach. As discussed in Section 3.3.5.4, the dataset is available in both CSV and PCAP formats. The first step in the proposed approach is to use the suitable and compatible CSV files that match the chosen DDoS attack vectors. In the approach, two chosen DDoS attacks TCP SYN and UDP are considered from 01-12 day.
For information about the chosen attacks see Table 6.3 [108].

Days	Attacks	Attack times	CSV Rows No.	PCAP files
01-12 Day	UDP	12 : 45 – 13 : 09	3136803	593 – 617
	SYN	13 : 29 – 13 : 34	1582682	618 – 818
	Benign	Other	–	189 – 191, 197 – 378, 471 – 474

Table 6.3 – CICDDoS2019 chosen attacks

6.2.1.1 Data Cleaning

Invalid or misleading data can pose a problem with either running the algorithm or the correct evaluation of the algorithm. Data cleaning is vital to guarantee correct interpretations of results and avoid real-time detection errors.

- **Rows**: The offered CSV files contain infinity values that cause errors while processing the data. The solution is to use Pandas python library [109] to replace the infinity values with Not a Number (NaN) values, then drop the rows that contain NaN values.

- **Columns**: As demonstrated before, CICDDoS2019 contains more than 80 features that was extracted using CICflowmeter [72] tool. Each column in the CSV files represents one feature.
  For dropping specific features, two methods are available:
  - **Unique Features**: This method can profit from the available analysis performed using **Radviz** [110] in [77] that contains the best five selected features that are relevant to each attack (except for MSSQL attack just two features).
    So the procedure is to drop all irrelevant or fuzzy features and leave only the distinct features for each attack.
  - **Valueless Features**: The option to drop all features except the best five selected features will lead to tiny images with [any height * any width], which negatively affects the performance of the VGG models. The reason behind this is that the default input shape for VGG16 and VGG19 is [224 height * 224 width] images, and any used image input should not be more minor than 32*32 [111]. The other possible option to drop random features and leave, for example, user-dependent features, such as socket features, leads to overfitting. Overfitting happens when training the model using user-dependent data like Flow ID, which will cause the model to learn in detail the given training data exceptionally well, which will negatively affect the ability of the models to deal with or recognize new data. The solution to tackling these two (tiny images and overfitting) problems is to use a method to drop only specific columns. These columns contain some relations between each other or between themselves—for example, dropping the following columns: highly correlated columns, constant columns, and zero-valued columns. Pandas profiling [112] have been used to execute this method.
    For full explanation of the warnings that distinct the columns in CICDDoS2019 dataset see Table 6.4 for 5000 rows of the TCP SYN CSV file. An overview about the TCP SYN CSV file see Figure 6.1.
    Pearson coefficient correlation can be used to determine the dependency between the columns using Equation (6.1) [113]. Figure 6.2 shows the Pearson coefficient correlation (Pearson's r) between the columns in the TCP SYN CSV file.

$$\rho_{X,Y} = \frac{E\left[\left(X-\mu_X\right)\left(Y-\mu_Y\right)\right]}{\sigma_X \sigma_Y} \tag{6.1}$$

Where σ_X and σ_Y denote the standard deviation of X and Y respectively and μ_X and μ_Y denote the expected values of X and Y respectively.

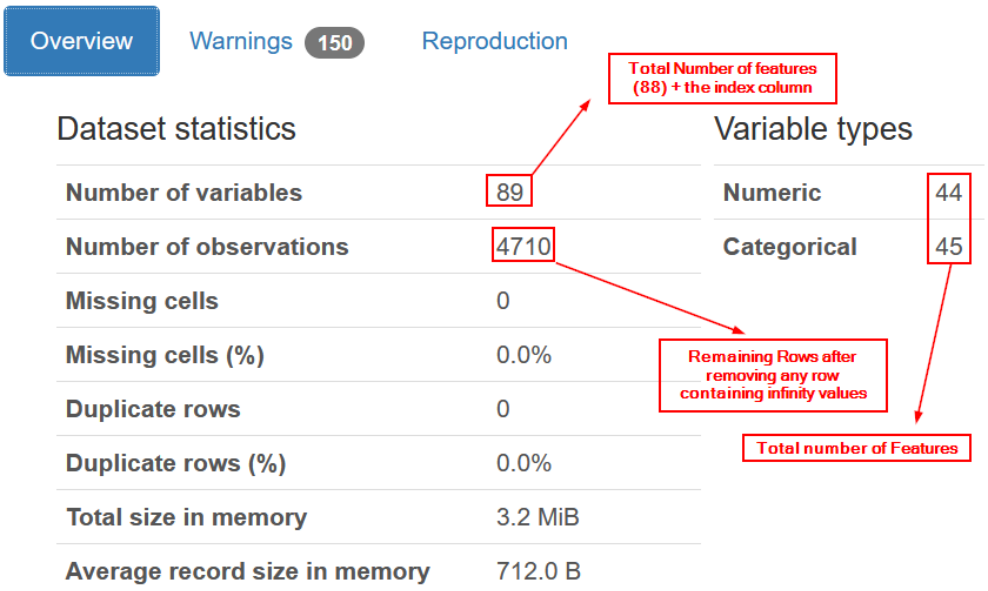


Figure 6.1 – Overview about 5000 rows in the SYNCSV file

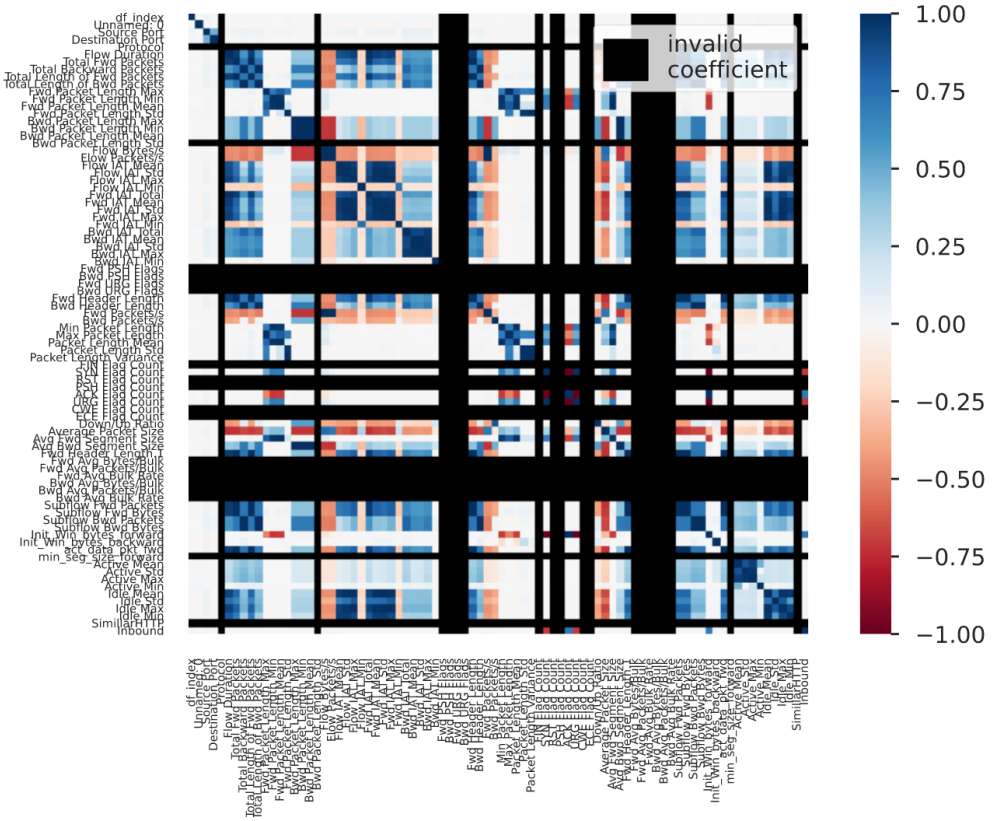


Figure 6.2 – Pearson correlation for SYN CSV file

Warning	Meaning	Features examples
High Cardinality	The columns with a high number of distinct values	Flow ID Timestamp
Constant	The columns with constant values	Protocol Label
High correlation	The columns with high correlation with other columns	ACK Flag Count Fwd Header Length
Skewed	Reflects the degree to which a standard bell-shaped probability distribution has been distorted	Active Min
Uniform	The columns with uniformly distribution values	Flow ID Timestamp
Unique	The columns that have unique values	Flow ID Unnamed:0 Timestamp
Zeros	The columns that contains a lot of zero values	Bwd IAT Std Active Std

Table 6.4 – Pandas profiling warnings for 5000 rows of the SYN CSV file [112]

Table 6.6 contains complete list of the possible features to be dropped for the chosen DDoS attacks vector. The decision to drop a feature is taken after taking into considerations all the warnings mentioned in Table 6.4 and after deep studying of the dataset using WEKA [114] tool as depicted in Figure 6.3 and Figure 6.4.
This analysis is done after considering the Pandas profiling analysis result and confirming the existence of the best five features mentioned in [77] for each attack as a unique feature. In addition to the 29 features mention in Table 6.6, the **Label** column will be dropped after splitting the data using it later as the real-time data is not labeled. Figure 6.5 shows the flowchart of the features selection process.

DDoS Attack	Features
TCP SYN	ACK Flag Count
	Init Win bytes forward
	min seg size forward
	Fwd IAT Total
	Flow Duration
UDP	Destination Port
	Fwd Packet Length Std
	Packet Length Std
	min seg size forward
	Protocol

Table 6.5 – Best 5 selected features for both TCP SYN & UDP DDoS attacks [77]

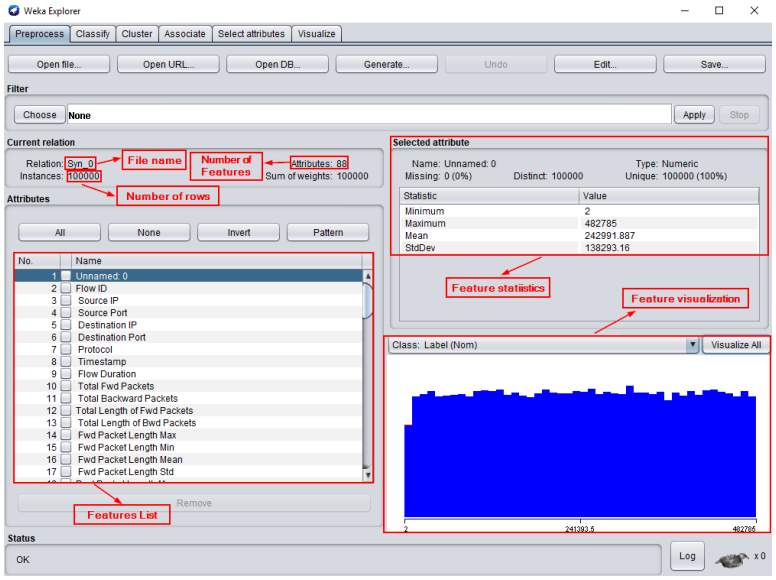


Figure 6.3 – Overview about WEKA tool dataset analysis

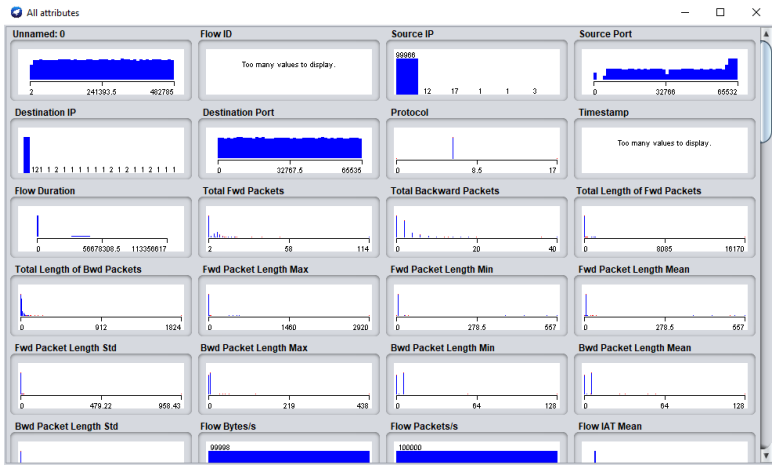


Figure 6.4 – Visualization of CICDDoS2019 features

Source IP	Destination IP	Source Port
Flow ID	Timestamp	Unnamed: 0
Active Max	Active Min	Active Std
URG Flag Count	Bwd Packet Length Std	Subflow Bwd Bytes
Bwd IAT Max	Subflow Bwd Packets	Bwd IAT Std
Idle Max	Idle Min	Fwd Bytes/Bulk Avg
Packet Length Mean	Packet Length Min	Fwd IAT Std
Bwd URG Flags	Bwd PSH Flags	Fwd Packet/Bulk Avg
Bwd Bytes/Bulk Avg	Fwd URG Flags	ECE Flag Count
Fwd Segment Size Avg	Bwd Segment Size Avg	

Table 6.6 – Features to drop from CICDDoS2019 dataset

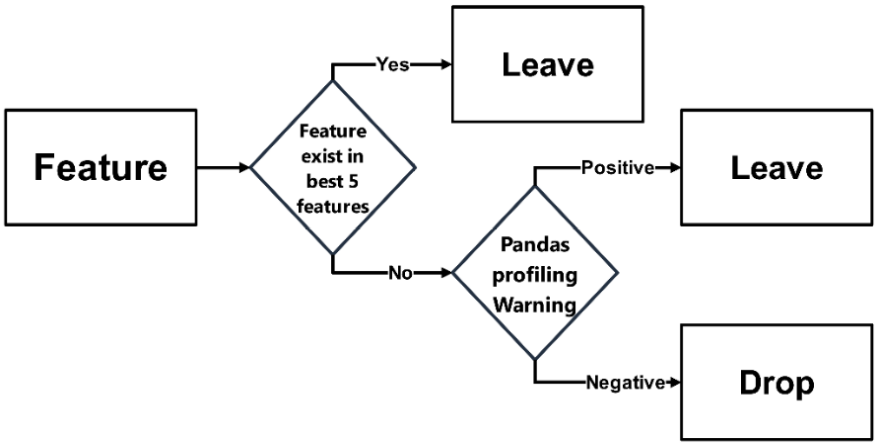


Figure 6.5 – Feature Selection Flowchart

6.2.1.2 Data Balancing

CICDDoS2019 is highly imbalanced dataset [115] [78]. This imbalanced structure is due to the high percentage of attack traffic compared to benign traffic.
After a deep analysis of the benign values/rows that exist in the whole dataset over the two chosen DDoS attacks (TCP SYN and UDP), it is noticeable that the percentage of the benign traffic (for only the first day 01-12 as in Table 6.3) is only **1.2**%.
The calculation of this value is done in two steps:

- **Benign traffic count**: Counting the benign labeled rows in all CSV files using the following command inside the windows CLI (for example for SYN):
  find /I /C "benign" Syn.csv, gives the number of benign rows in the SYN CSV file.

- **Total traffic count**: Counting the total rows number in both the SYN CSV file and the UDP CSV file using the following two commands inside the windows CLI:
  find /V /C "anytext" Syn.csv, gives the number of rows in the SYN CSV file.
  find /V /C "anytext" UDP.csv, gives the number of rows in the UDP CSV file.

The following are the three solutions to this problem (try and error method to find the best or even a combination of two of them):

- **Sampling**: The idea is either reducing the attack traffic weight compared to benign or increasing the benign traffic weight compared to the attack traffic—the implementation by simply choosing a specific number of images as needed to achieve balanced ratio.
- **CICFlowmeter**: Using CICFlowmeter tool to convert the given benign PCAP files into CSV files that contain the traffic in addition to the features, then convert these CSV files into images.
- **CICIDS2017**: The CICIDS2017 dataset is containing one whole benign traffic CSV file under the name **Monday-WorkingHours.pcap_ISCX** with total of 176400 benign rows.

After trying the three solutions for assessment, the chosen solution is the second solution.

6.2.2 Images

As mentioned earlier in Section 4.2, using images generally increases the performance of the general version of deep learning neural networks like CNN and the specific pre-trained architectures like VGG, Inception, AlexNet, LeNet, or ResNet.
The performance improvement is because they are initially examining the visual imagery. Inspired by the fact that having a graphic version of a dataset motivates and enables new ideas to utilize, exploit, and integrate the dataset as much as it can be.
For example, lots of deep analysis and improvements was done for available visual dataset like **Malimg** dataset Figure 6.6 to detect malware [116], [117] and [118]. As shown in Figure 6.8, the original available visual malimg dataset was in grayscale images. Researchers in [119] proved that by using Deep Convolutional Generative Adversarial Network (DCGAN) to generate a colored version of malimg dataset Figure 6.7, an improvement in malware classification is achieved.
Other examples, making available image datasets like Diabetic Retinopathy [120], Pneumonia [121], and many other medical image datasets, helped improve prognosis and diagnose the human body's anomalies using deep learning algorithms.
To better use the CICDDoS2019 dataset using deep learning models, a graphic version is

needed. There is no online available visual version of it to the best of my knowledge. Other researchers tried to provide a visual version of CICDDoS2019 in IoT [122].

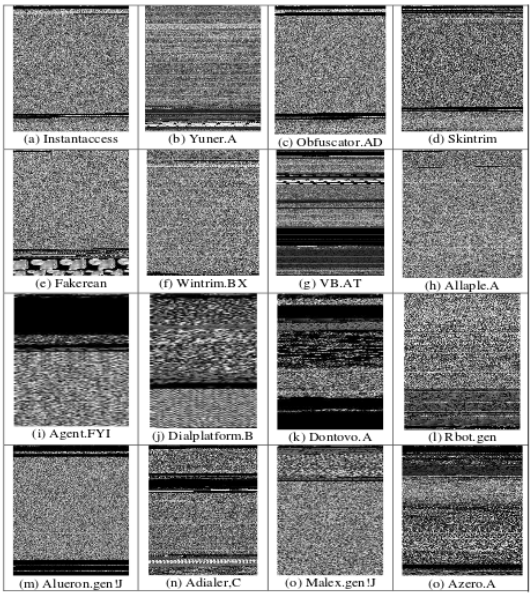


Figure 6.6 – Grayscale [123]

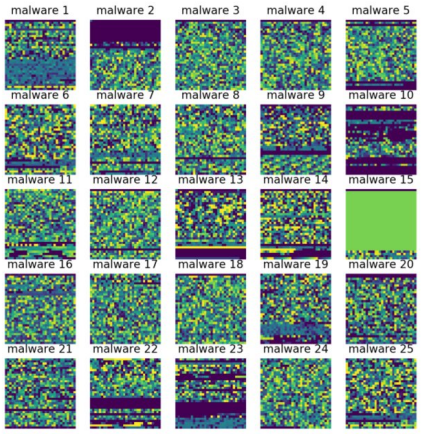


Figure 6.7 – Colored [119]

Figure 6.8 – Malimg dataset and the colored version of it

6.2.2.1 Data Cleaning

As mentioned in Section 6.2.1.1, data cleaning is needed to filter the dataset files from any row containing infinity or NaN values. In addition, it is also dropping all columns that have valueless features.

6.2.2.2 Splitting the dataset

Generally, two dataset splitting is needed. The first is splitting the dataset files into benign or attack traffic to convert the dataset into images. The second is splitting the resultant images to train and test sets to be processed in the deep learning model.

- **CICDDoS2019 traffic split**: The creation of a new visual version of CICDDoS2019 dataset needs a split of the dataset, so classified images either to a **benign** category or an **attack** category are available. Splitting the dataset is done by benefiting from the fact that the CICDDoS2019 dataset is a labeled dataset, so just gathering the Benign-labeled or the non-benign-labeled traffic into specific data frames will result in a categorized visual dataset.

- **CICDDoS2019 images split**: Standard 70:30 split is used to split the resultant images dataset into train and test sets.

6.2.2.3 Normalize the dataset

Dataset normalization (feature range) between (0,255) using **MinMaxScaler** is needed to avoid considerable training time.

6.2.2.4 Dataset batches

Dividing the resultant categorized CSV (Rows*Columns = Rows*58) files into 174 rows batches. Every batch internally will be divided into three internal batches; every batch is 58 rows*58 columns. The first batch is considered as the **Red** channel, the second batch is considered as the **Green** channel, and the third and last channel is considered as the **Blue** channel.

6.2.2.5 Channels accumulation

Any colored image consists of three channels (R,G,B) in our case - there are other basic colors representations like Cyan Magenta Yellow Black (CMYK) - and adding the three layers together results in a colored image with a size equal to 58 pixels*58 pixels.

The full flowchart of the image process is depicted in Figure 6.9.

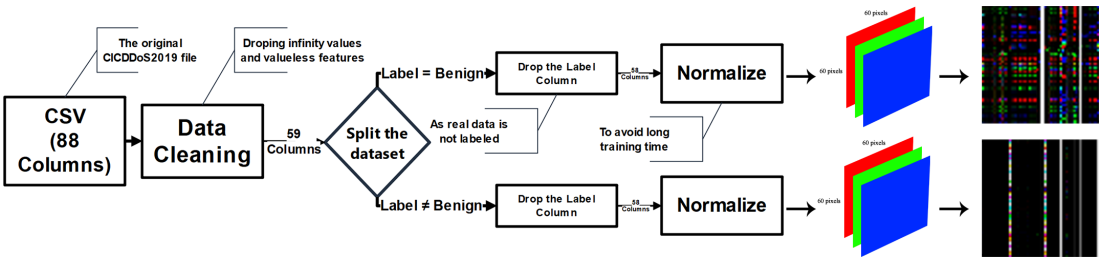


Figure 6.9 – CICDDoS2019 Images dataset creation process

A given sample about the resultant CICDDoD2019 images datasets is shown in Figure 6.10.

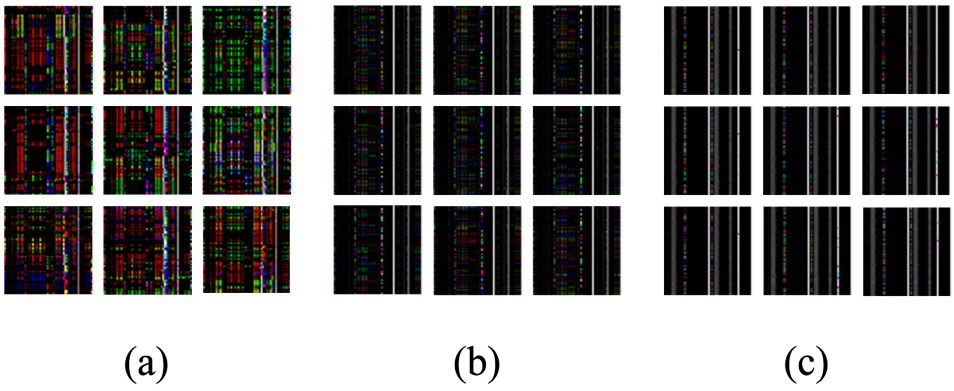


Figure 6.10 – Samples of CICDDoS2019 Images dataset (a) Benign (b) SYN (c) UDP

6.2.3 VGG16

VGG [124] is a pre-trained model which was trained on the ImageNet [125] dataset. Imagenet is a huge visual dataset with more than 14 million images created, especially for identifying visual objects. The idea behind the name VGG16 is that the number of layers in this model is 16 layers.
The advantages of using a pre-trained model is as follows:

- **Large classes variations Support**: Pre-trained models have been trained using 1000 classes, giving it the ability to distinguish between a large number of classes.
- **Time-saving**: Training a model is time-consuming, requiring so much time.
- **Easily Usable**: Calculations for a deep learning model are complex. For example, the number of layers? Size of filter? The number of nodes per layer? Size of pooling layer? Etc. The process is a pre-trained model is much easier as all those values are predefined and known for best matching.

The architecture of VGG16 is shown in Figure 6.11.

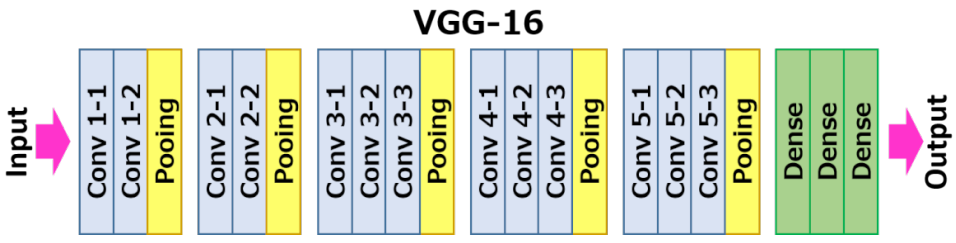


Figure 6.11 – VGG16 Architecture [126]

6.2.4 CICFlowmeter

CICFlowMeter [72] is a bidirectional network traffic analyzer and generate and it is offered in both CLI and GUI versions. As shown in Figure 6.12, it can work in online or realtime modes. The online mode means capturing the traffic from a specific network card. The offline mode means it can convert the already exist PCAP file into a CSV containing the previously mentioned features in Table 3.3. For more information about CICFlowMeter tool it is discussed in [127], [128].

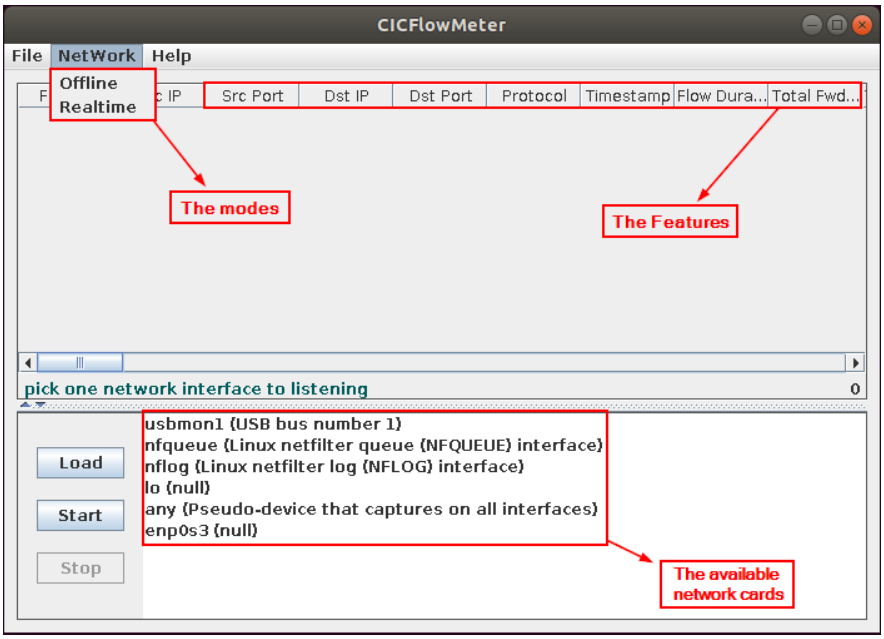


Figure 6.12 – CICFlowMeter GUI

Chapter 7

Simulation Results

7.1 SDN Topology

As shown in Figure 7.1, for the proposed model to be tested, a SDN topology architecture is needed, in which multi-controllers are used to achieve high availability and overcome the SPOF problem. The topology consists of two controllers, two OpenFlow switches and four hosts.

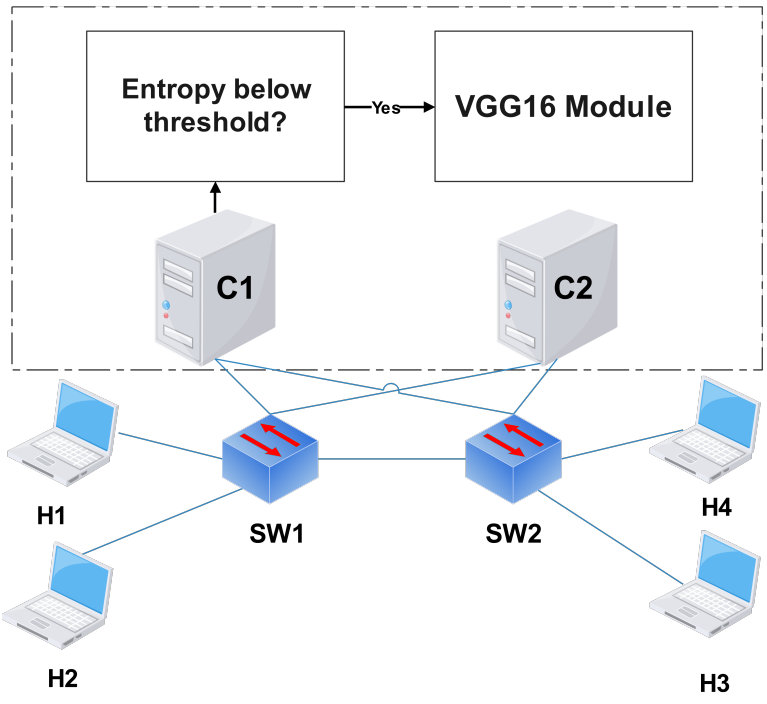


Figure 7.1 – The SDN topology testbed setup

7.1.1 Traffic generation

First of all, to simulate the real traffic generation scenario, the steps as follows:

- **IPs Choose:** For the attacker and the victiom, the IPs for both will be chosen based on the IPs in the PCAP file.
- **Xterm session:** Open xterm session on **H1**.
- **Launch the DDoS attack: H1** will generate a DDoS traffic using **TCPreplay** command, for example, **tcpreplay -i h1-eth0 -v -tK –loop 5000 –unique-ip PCAP_File_NAME**

7.1.2 Detection Phase

To detect a DDoS attack, two phases are used to ensure high accuracy and low controller overload:

7.1.2.1 Entropy Module

The main target of the proposed approach is to detect and mitigate the DDoS attacks. Continuously converting every set of packets to an image overloads the model; an introductory module is needed to analyze the traffic characteristics. The entropy module represents the first step to identify the traffic. Based on using different flow identifications, different results show up. In our case, the destination IPv4 address is the sole tuple. As depicted in Figure 7.2, after further analysis of using a window size of 174 packets, the chosen threshold value is 0,5.

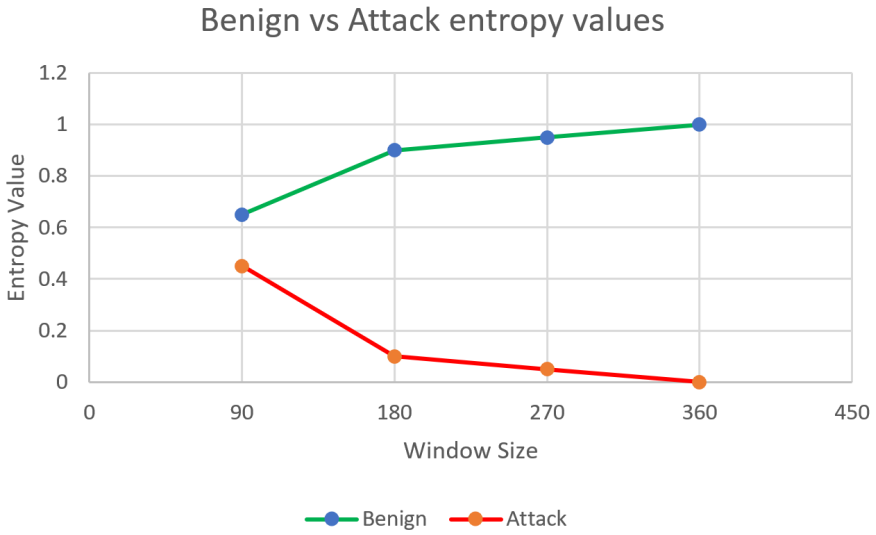


Figure 7.2 – Entropy values for different window sizes for both benign and attack traffics

7.1.2.2 VGG16 Module

If the entropy value of the traffic is below the preset threshold, the next step is to activate the VGG16 model to analyze the traffic further. In this module, two possibilities can exist as follows:

- **Binary classification:** It is a two-class classification to classify the traffic either as benign or attack. The used activation function in such classification is the **sigmoid** activation function as shown in Equation (7.1).

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{7.1}$$

- **Categorical classification:** It is a Multi-class classification (one of more than two classes) to classify the traffic either as benign or one of more than one DDoS attack. The used activation function in such classification is the **softmax** activation function as shown in Equation (7.2).

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1,2,\ldots,K \tag{7.2}$$

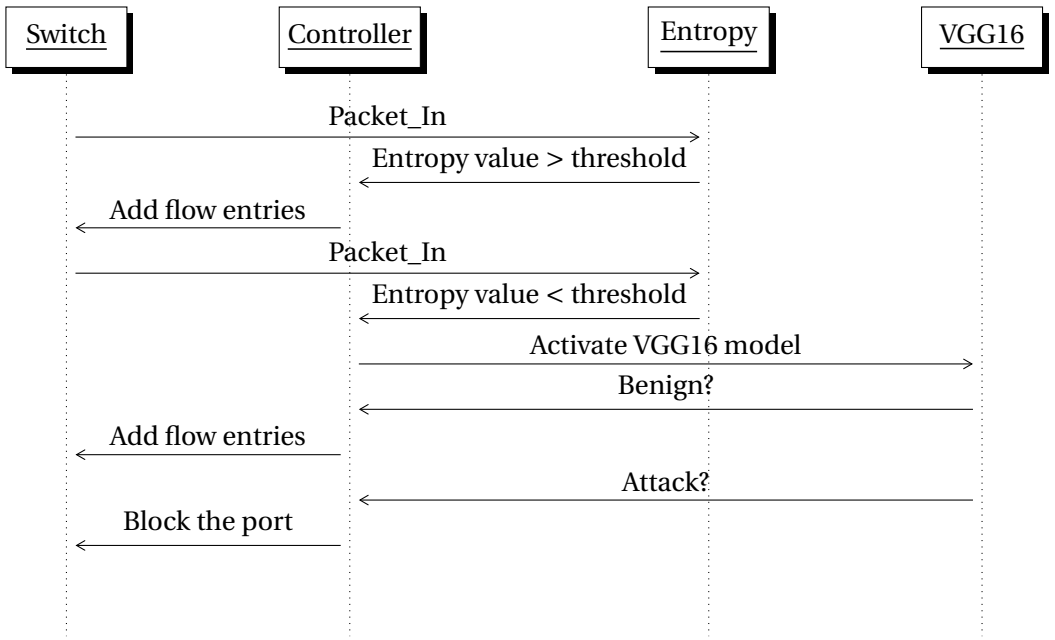
7.1.3 Mitigation Phase

For the mitigation part and as shown in the above messages sequence chart, blocking the device's port launching the attack is the chosen way of mitigation.
Blocking the port of the attacker is executed using two OpenFlow switches, **OFPT_FLOW_-MOD** and **OFPT_PORT_MOD**, which are two controller-to-switch messages as explained in Section 2.2.2.
The **OFPT_PORT_MOD** [129] messages have many parameters, most important in our scenario as follows:

- **Port_no:** The unique identifier of the port.
- **Config:** the updated value of specific fields.
- **Mask:** contains the fields that the config values will replace.

7.1.4 Modules Inter-connections

The connections between different modules and the controllers are described in the following messages sequence chart:



7.2 Evaluation Metrics

The commonly accepted metrics in anomaly detection to evaluate the proposed approach are as follows:

- **True Positive (TP):** Attacks are rightly classified as attacks.
- **False Positive (FP):** Benigns are falsely classified as attacks.
- **True Negative (TN):** Benigns are rightly classified as benigns.
- **False Negative (FN):** Attacks are falsely classified as benigns.

7.2.1 Accuracy

Overall review of the approach to evaluate its performance.

$$Accuracy = \frac{TP + TF}{TP + TF + FP + FN} \tag{7.3}$$

7.2.2 Precision

The accuracy percentage of the true positive (attacks/anomalies) classifications over the all positive (attacks/anomalies) classifications. The higher the better.

$$Precision = \frac{TP}{TP + FP} \tag{7.4}$$

7.2.3 Recall (Sensitivity)

The accuracy percentage of the true positive (attacks/anomalies) classifications over the actual positive (attacks/anomalies) values. The higher the better.

$$Recall = \frac{TP}{TP + FN} \tag{7.5}$$

7.2.4 F1 Score

The F1 Score is a trade-off between precision and recall values.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{7.6}$$

Worth mentioning that **precision** and **recall** values are the most prominent in attacks detections [130]. The reason behind that is that both values measure the actual positive classified attacks.

All mentioned previous values in Section 7.2 can be used to plot the confusion matrix. **The confusion matrix** measures how much the model is confused, which is visually helpful to assess the model. The confusion matrices for both binary classifications and Categorical classifications (for three classes) are depicted in Figure 7.5.

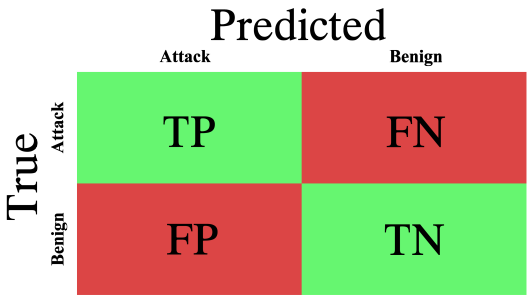


Figure 7.3 – Binary

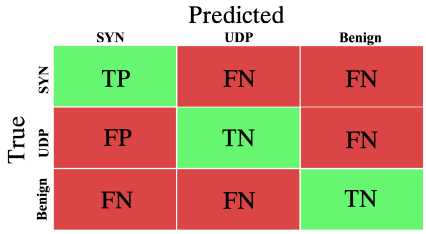


Figure 7.4 – Categorical

Figure 7.5 – Confusion matrix for Binary and Categorical classification

7.2.5 Results

For the proposed model (VGG16) to detect DDoS attacks in SDN, the following are the achieved results for the binary classification scenario:

- **True Positive that is Predicted Positive=** 63692
- **True Positive that is Predicted Negative=** 453

- **True Negative that is Predicted Positive=** 264
- **True Negative that is Predicted Negative=** 49762

Models	Accuracy %	Precision %	Recall %	F1 %
ID3 [77]	-	0.78	0.65	0.69
RF [77]	-	0.77	0.56	0.62
Naïve Bayes [77]	-	0.41	0.11	0.05
LR [77]	-	0.25	0.02	0.04
CNN 3C3F [88]	0.9906	0.9905	0.9908	0.9906
Proposed model	0.9937	0.9959	0.9929	0.9944

Table 7.1 – Evaluation metrics of six models

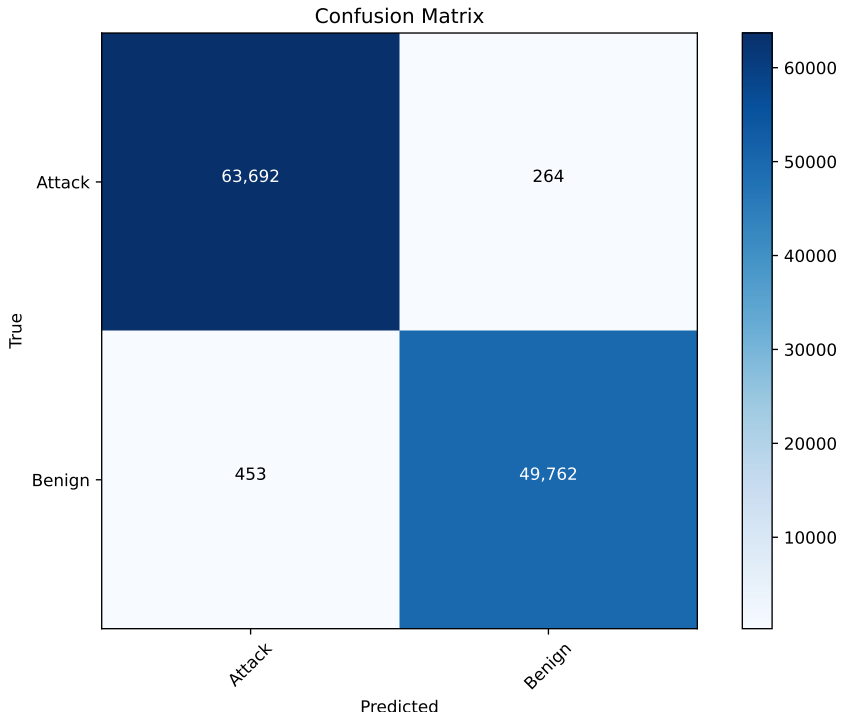


Figure 7.6 – The confusion matrix of the binary classification case

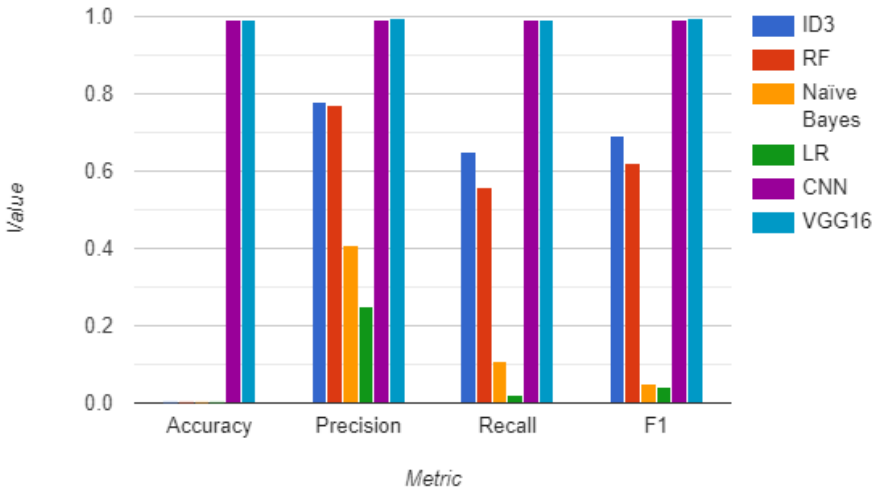


Figure 7.7 – Evaluation metrics of six models

For categorical classification, the confusion matrix for the results as shown in Figure 7.8:

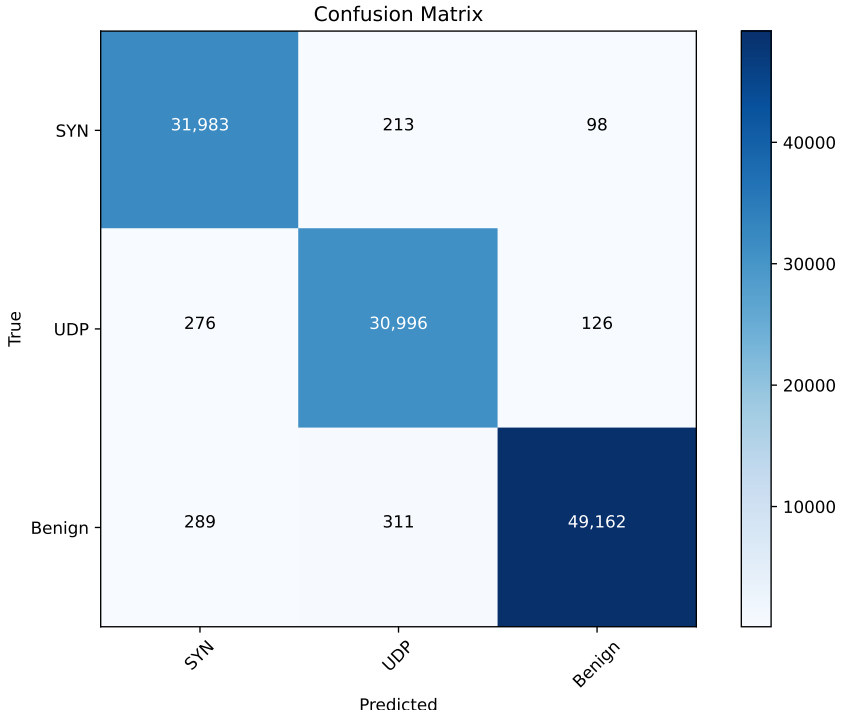


Figure 7.8 – The confusion matrix of the categorical classification case

7.2.6 Improvements from state-of-the-art works

As shown in Table 7.1 and Figure 7.7, the proposed approach of using VGG16 as the detection module, outperformed the state-of-the-art [77] and [88] in detecting DDoS attacks in SDN. The more distinguished values are due to the benefit of using the introduced visual version of the most modern DDoS attacks datasets (CISDDoS2019) as an input for a pre-trained very deep learning model VGG16.
In [77], machine learning models were used. In addition, in [88], although using CNN deep learning model, this is not a very deep learning model, and the used method to generate the traffic is a tool (Hping3) instead of using real PCAP files as a source of DDoS attacks.

Chapter 8

Conclusion and Future work

8.1 Conclusion

Based on the proposed approach in this thesis, using a transfer learning model like VGG16 outperforms the abstract CNN model in identifying and classifying DDoS attacks. The main contribution of this thesis is using an alternative way (Images) to process the most modern DDoS attacks dataset CICDDoS2019 in SDN. The main focus was to prove that using colored images as an input for a transfer learning model can significantly improve DDoS attacks detection. This improvement can encourage companies to adopt and test the proposed approach from an experimental perspective.

8.2 Future work

There is an urgent need for further deep analysis in the area of DDoS attacks in SDN due to many concepts to be considered. The following ideas could be taken into consideration:

1. Investigations and comparisons should cover the other types of DDoS attacks using the other transfer learning models, for example, VGG19, AlexNet, and ResNet50.
2. Instead of using one standard deep learning algorithm, combining two or more deep learning algorithms could help to get better results.
3. Adjusting image size/channels to find the best usable image size/channel system to provide the highest evaluation metrics values.
4. Find a way to identify traffic of two or more simultaneous DDoS attacks. For example, identifying simultaneous TCP SYN and UDP attacks.

List of Acronyms

ACK	Acknowledgement
ACL	Access Control List
AI	Artificial Intelligence
API	Application Programming Interface
APIs	Application Programming Interfaces
AWS	Amazon Web Services
BGP	Border Gateway Protocol
BPNN	Back Propagation Neural Network
BR	Byte Rate
C&C	Command and Control
CIA	Confidentiality, Integrity and Availability
CIC	Canadian Institute for Cybersecurity
CISC	Complex Instruction Set Computing
CLI	Command-Line Interface
CMYK	Cyan Magenta Yellow Black
CNN	Convolutional Neural Network
CRUD	Create, Read, Update and Delete
CSE	Communications Security Establishment
CSV	Comma-Separated Values
DCGAN	Deep Convolutional Generative Adversarial Network
DDoS	Distributed Denial of Service
DNS	Domain Name Server
DoS	Denial of Service

FC	Fully-Connected
FFT	Fast Fourier Transform
FN	False Negative
ForCES	Forwarding and Control Element Separation
FP	False Positive
FPSA	Flows Percentage with Small Amount Packets
FTP	File Transfer Protocol
GCP	Google Cloud Platform
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
ID3	Iterative Dichotomiser 3
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Internet Protocol
IPv6	Internet Protocol version 6
ISPs	Internet Service Providers
KNN	K-Nearest Neighbors
LISP	Locator/ID Separation Protocol
LOIC	Low Orbit Ion Canon
LR	Linear Regression
MPLS	Multiprotocol Label Switching
NB	Naive Bayes
ODM	Original Device Manufacturer
OSI	Open Systems Interconnection
OXM	OpenFlow Extensible Match
QoS	Quality Of Service

PCAP	Packet Capture
PCAPNG	Packet Capture Next Generation
R2L	Remote to Local
Rest	Representational state transfer
RF	Random Forest
RGB	Red Green Blue
RISC	Reduced Instruction Set Computer
RNN	Recurrent Neural Network
SDN	Software-Defined Networking
sFLOW	Sampled FLOW
SFP	Symmetric Flows Percentage
SGS	Safe-Guard Scheme
SNMP	Simple Network Management Protocol
SPOF	Single Point Of Failure
SSH	Secure Shell
SVM	Support Vector Machine
SYN	Synchronize
TCAM	Ternary Content-Addressable Memory
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
TLS	Transport Layer Security
MITM	Man-In-The-Middle
NaN	Not a Number
U2R	User to Root
UDP	User Datagram Protocol
VAFR	Variation Rate of Asymmetric Flow
VGG	Visual Geometry Group
VLAN	Virtual Local Area Network
VRRP	Virtual Router Redundancy Protocol

List of Figures

List of Tables

Bibliography

[1] C. A. I. Report, "global population internet access 2018-2023," March 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf

[2] J. Alcober, X. Hesselbach, A. de la Oliva, A. Garcia-Saavedra, D. Roldan, and C. Bock, "Internet future architectures for network and media independent services and protocols," in *2013 15th International Conference on Transparent Optical Networks (ICTON)*, 2013.

[3] T. D. Nadeau and K. Gray, in *SDN: Software Defined Networks*, 2013.

[4] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A sdn-oriented ddos blocking scheme for botnet-based attacks," in *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2014.

[5] G. Pujolle, in *Software Networks : Virtualization, SDN, 5G, and Security*, vol. 1, 2020.

[6] D. P. A. Brij B. Gupta, Gregorio Martinez Perez and D. Gupta, in *Handbook of Computer Networks and Cyber Security*, 2020.

[7] C. B. Paul Göransson and T. Culver, in *Software Defined Networks A Comprehensive Approach*, 2017.

[8] Risc vs. cisc architectures. https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/.

[9] T. H. P. T.-G. Thomas Zinner, Michael Jarschel and W. Kellerer, "A compass through sdn networks," December 2013. [Online]. Available: https://mediatum.ub.tum.de/doc/1200415/1200415.pdf

[10] Requirements for separation of ip control and forwarding. https://tools.ietf.org/html/rfc3654.

[11] Forwarding and control element separation (forces) framework. https://tools.ietf.org/html/RFC3746.

[12] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic, "A survey on data plane flexibility and programmability in software-defined networking," *IEEE Access*, vol. 7, pp. 47 804–47 840, 2019.

[13] Sdn architecture. https://opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf.

[14] S. Jouet and D. P. Pezaros, "Bpfabric: Data plane programmability for software defined networks," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2017, pp. 38–48.

[15] K. Kim, S. Min, and Y. Han, "A programmable data plane to support in-network data processing in software-defined iot," in *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, 2017, pp. 855–860.

[16] E. Kaljic, A. Maric, and M. Hadzialic, "A novel qualitative metric based approach to the improvement of data plane flexibility in software-defined networks," in *IEEE EUROCON 2019 -18th International Conference on Smart Technologies*, 2019.

[17] Q. Yan and F. R. Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 52–59, 2015.

[18] C. Bäumlisberger, "Plattformunabhängige lokale sdn-controller auf offener weiterleitungshardware durch anwendung von containertechnologie," 2016. [Online]. Available: http://elib.uni-stuttgart.de/handle/11682/9421

[19] H. G. Ahmed and R. Ramalakshmi, "Performance analysis of centralized and distributed sdn controllers for load balancing application," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2018, pp. 758–764.

[20] O. Blial, M. Ben Mamoun, and R. Benaini, "An overview on sdn architectures with multiple controllers," *J. Comput. Netw. Commun.*, vol. 2016, Apr. 2016. [Online]. Available: https://doi.org/10.1155/2016/9396525

[21] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, "A comprehensive survey of interface protocols for software defined networks," *Journal of Network and Computer Applications*, vol. 156, p. 102563, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520300370

[22] A. Rodriguez-Natal, M. Portoles-Comeras, V. Ermagan, D. Lewis, D. Farinacci, F. Maino, and A. Cabellos-Aparicio, "Lisp: a southbound sdn protocol?" *IEEE Communications Magazine*, vol. 53, no. 7, 2015.

[23] N. Anerousis, P. Chemouil, A. A. Lazar, N. Mihai, and S. B. Weinstein, "The origin and evolution of open programmable networks and sdn," *IEEE Communications Surveys Tutorials*, 2021.

[24] Sdn architecture. https://tools.ietf.org/html/draft-yin-sdn-sdni-00.

[25] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: https://doi.org/10.1145/1355734.1355746

[26] O. N. Foundation, "Openflow switch specification 1.3.5," March 2015. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.3.5.pdf

[27] S. Muhizi, G. Shamshin, A. Muthanna, R. Kirichek, A. Vladyko, and A. Koucheryavy, "Analysis and performance evaluation of SDN queue model," in *Lecture Notes in Computer Science*. Springer International Publishing, 2017, pp. 26–37. [Online]. Available: "https://doi.org/10.1007/978-3-319-61382-6_3"

[28] D. O'Briain, "Ryu sdn [] testbed manual," 2019. [Online]. Available: http://www.obriain.com/training/sdn/RYU_Soft_Testbed_v1.6_odt.pdf

[29] B. Isyaku, M. S. M. Zahid, M. B. Kamat, K. A. Bakar, and F. A. Ghaleb, "Software defined networking flow table management of OpenFlow switches performance and security challenges: A survey," *Future Internet*, vol. 12, no. 9, p. 147, Aug. 2020. [Online]. Available: https://doi.org/10.3390/fi12090147

[30] O. N. Foundation, "Openflow switch specification 1.1," December 2009. [Online]. Available: https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf

[31] ——, "Openflow switch specification 1.2," December 2011. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.2.pdf

[32] ——, "Openflow switch specification 1.3," June 2012. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[33] ——, "Openflow switch specification 1.5.1," March 2015. [Online]. Available: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[34] 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. [Online]. Available: https://doi.org/10.3390/fi6020302

[35] Ching-Hao and D. Y. Lin, "Openflow version roadmap."

[36] F. Bannour, S. Souihi, and A. Mellouk, "Distributed sdn control: Survey, taxonomy, and challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 333–354, 2018.

[37] C. Yoon, S. Lee, H. Kang, T. Park, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Flow wars: Systemizing the attack surface and defenses in software-defined networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3514–3530, 2017.

[38] M. Mahdy, "Sdn southbound threats," September 2018. [Online]. Available: https://www.sans.org/reading-room/whitepapers/networksecurity/sdn-southbound-threats-38700

[39] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[40] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[41] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*. ACM Press, 2013. [Online]. Available: https://doi.org/10.1145/2491185.2491199

[42] M. Antikainen, T. Aura, and M. Särelä, "Spook in your network: Attacking an SDN with a compromised OpenFlow switch," in *Secure IT Systems*. Springer International Publishing, 2014, pp. 229–244. [Online]. Available: https://doi.org/10.1007/978-3-319-11599-3_14

[43] F. M. D. M. R. D. F. A. Maham Iqbal, Farwa Iqbal, "Security issues in software defined networking (sdn): Risks, challenges and potential solutions," October 2019. [Online]. Available: https://thesai.org/Downloads/Volume10No10/Paper_42-Security_Issues_in_Software_Defined_Networking.pdf

[44] J. Benabbou, K. Elbaamrani, and N. Idboufker, "Security in OpenFlow-based SDN, opportunities and challenges," *Photonic Network Communications*, vol. 37, no. 1, pp. 1–23, Oct. 2018. [Online]. Available: https://doi.org/10.1007/s11107-018-0803-7

[45] Y. H. XU Xiaoqiong and Y. Kun, "Ddos attack in software defined networks: A survey," September 2017. [Online]. Available: https://res-www.zte.com.cn/mediares/magazine/publication/com_en/article/201703/465747/P020171018323522738119.pdf

[46] L. F. Eliyan and R. D. Pietro, "DoS and DDoS attacks in software defined networks: A survey of existing solutions and research challenges," *Future Generation Computer Systems*, Mar. 2021. [Online]. Available: https://doi.org/10.1016/j.future.2021.03.011

[47] N. A. Aziz, T. Mantoro, M. A. Khairudin, and A. F. b. A. Murshid, "Software defined networking (sdn) and its security issues," in *2018 International Conference on Computing, Engineering, and Design (ICCED)*, 2018.

[48] A. Pradhan and R. Mathew, "Solutions to vulnerabilities and threats in software defined networking (SDN)," *Procedia Computer Science*, vol. 171, pp. 2581–2589, 2020. [Online]. Available: https://doi.org/10.1016/j.procs.2020.04.280

[49] N. Katta, O. Alipourfard, J. Rexford, and D. Walker, "Infinite CacheFlow in software-defined networks." ACM, Aug. 2014. [Online]. Available: https://doi.org/10.1145/2620728.2620734

[50] I. E. F. Tulio A. Pascoal, Yuri G. Dantas and V. Nigam, "Slow tcam exhaustion ddos attack," May 2017. [Online]. Available: http://download.hrz.tu-darmstadt.de/pub/FB20/Dekanat/Publikationen/Crossing/ifipsec17-camera-ready.pdf

[51] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense," *Security and Communication Networks*, vol. 2018, pp. 1–15, 2018. [Online]. Available: https://doi.org/10.1155/2018/4760632

[52] S. Ali, M. K. Alvi, S. Faizullah, M. A. Khan, A. Alshanqiti, and I. Khan, "Detecting ddos attack on sdn due to vulnerabilities in openflow," in *2019 International Conference on Advances in the Emerging Computing Technologies (AECT)*, 2020, pp. 1–6.

[53] S. Hommes, R. State, and T. Engel, "Implications and detection of dos attacks in openflow-based networks," in *2014 IEEE Global Communications Conference*, 2014, pp. 537–543.

[54] R. Klöti, V. Kotronis, and P. Smith, "Openflow: A security analysis," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, 2013, pp. 1–6.

[55] ccexpert, "Introducing the macof tool," March 2021. [Online]. Available: https://www.ccexpert.us/port-security/introducing-the-macof-tool.html

[56] A. Sebbar, M. Boulmalf, M. Dafir Ech-Cherif El Kettani, and Y. Baddi, "Detection mitm attack in multi-sdn controller," in *2018 IEEE 5th International Congress on Information Science and Technology (CiSt)*, 2018, pp. 583–587.

[57] C. Banse and S. Rangarajan, "A secure northbound interface for SDN applications," in *2015 IEEE Trustcom/BigDataSE/ISPA*. IEEE, Aug. 2015. [Online]. Available: https://doi.org/10.1109/trustcom.2015.454

[58] E. Džaferović, A. Sokol, A. A. Almisreb, and S. M. Norzeli, "DoS and DDoS vulnerability of IoT: A review," *Sustainable Engineering and Innovation*, vol. 1, no. 1, pp. 43–48, Jun. 2019. [Online]. Available: https://doi.org/10.37868/sei.v1i1.36

[59] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[60] S. Z. Tajalli, M. Mardaneh, E. Taherian-Fard, A. Izadian, A. Kavousi-Fard, M. Dabbaghjamanesh, and T. Niknam, "DoS-resilient distributed optimal scheduling in a fog supporting IIoT-based smart microgrid," *IEEE Transactions on Industry Applications*, vol. 56, no. 3, pp. 2968–2977, May 2020. [Online]. Available: https://doi.org/10.1109/tia.2020.2979677

[61] Radware, "Ddos survival handbook," 2013. [Online]. Available: https://www.radware.com/getattachment/Security/Research/702/Radware_DDoS_Handbook_2015.pdf.aspx?lang=en-US

[62] embeddedcomputing, "The history and evolution of ddos attacks," October 2020. [Online]. Available: https://www.embeddedcomputing.com/technology/security/network-security/the-history-and-evolution-of-ddos-attacks

[63] D. Menscher, "Exponential growth in ddos attack volumes," October 2020. [Online]. Available: https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks

[64] T. L. R. Q. 2020, "Aws shield," 2020. [Online]. Available: https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf

[65] D. K. B. J. K. Kalita, "Ddos attacks evolution, detection, prevention, reaction, and tolerance," 2016.

[66] securelist.com, "Ddos attacks in q4 2020," February 2021. [Online]. Available: https://securelist.com/ddos-attacks-in-q4-2020/100650/

[67] allot.com, “Ddos attack handbook,” 2018. [Online]. Available: https://allot.com/docs/ddos-attack-handbook.pdf

[68] U. of Applied Sciences Fulda, “Ndsec-1 dataset,” 2017. [Online]. Available: https://www2.hs-fulda.de/NDSec/NDSec-1/Files/

[69] F. Beer, T. Hofer, D. Karimi, and U. Bühler, “A new attack composition for network security,” in *10. DFN-Forum Kommunikationstechnologien*, P. Müller, B. Neumair, H. Raiser, and G. Dreo Rodosek, Eds. Bonn: Gesellschaft für Informatik e.V., 2017, pp. 11–20.

[70] F. Beer and U. Bühler, “Feature selection for flow-based intrusion detection using rough set theory,” in *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*, 2017, pp. 617–624.

[71] C. I. for Cybersecurity, “Intrusion detection evaluation dataset (cic-ids2017),” 2017. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html

[72] ——, “Cicflowmeter,” 2017. [Online]. Available: https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter

[73] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, and R. Budiarto, “Cicids-2017 dataset feature analysis with information gain for anomaly detection,” *IEEE Access*, vol. 8, pp. 132 911–132 921, 2020.

[74] C. I. for Cybersecurity, “Cse-cic-ids2018 on aws,” 2018. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[75] J. L. Leevy and T. M. Khoshgoftaar, “A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 big data,” *Journal of Big Data*, vol. 7, no. 1, Nov. 2020. [Online]. Available: https://doi.org/10.1186/s40537-020-00382-x

[76] C. I. for Cybersecurity, “Ddos evaluation dataset (cic-ddos2019),” 2019. [Online]. Available: https://www.unb.ca/cic/datasets/ddos-2019.html

[77] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “Developing realistic distributed denial of service (ddos) attack dataset and taxonomy,” in *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, pp. 1–8.

[78] J. LI, “Detection of ddos attacks based on dense neural networks, autoencoders and pearson correlation coefficient,” 2020. [Online]. Available: http://hdl.handle.net/10222/78536

[79] S. M. Mousavi and M. St-Hilaire, "Early detection of ddos attacks against sdn controllers," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, 2015, pp. 77–81.

[80] P. Wiki, "Pox." [Online]. Available: http://intronetworks.cs.luc.edu/auxiliary_files/mininet/poxwiki.pdf

[81] [Online]. Available: https://doi.org/10.18280/rces.060201

[82] B. U. Tamer Omar, Anthony Ho, "Detection of ddos in sdn environment using entropy-based detection," in *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2019, pp. 1–4.

[83] K. S. Sahoo, B. Sahoo, M. Vankayala, and R. Dash, "Detection of control layer ddos attack using entropy metrics in sdn: An empirical investigation," in *2017 Ninth International Conference on Advanced Computing (ICoAC)*, 2017, pp. 281–286.

[84] R. Li and B. Wu, "Early detection of ddos based on φ-entropy in sdn networks," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1, 2020, pp. 731–735.

[85] [Online]. Available: https://doi.org/10.1016/j.comnet.2017.02.015

[86] S. K. Yadav, P. Suguna, and R. L. Velusamy, "Entropy based mitigation of distributed-denial-of-service (ddos) attack on control plane in software-defined-network (sdn)," in *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2019, pp. 1–7.

[87] X. Yuan, C. Li, and X. Li, "Deepdefense: Identifying ddos attack via deep learning," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2017, pp. 1–8.

[88] L. Wang and Y. Liu, "A ddos attack detection method based on information entropy and deep learning in sdn," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1, 2020, pp. 1084–1088.

[89] [Online]. Available: https://doi.org/10.1155/2018/9804061

[90] S. Sanfilippo, "hping3." [Online]. Available: http://www.hping.org/

[91] S. S. Mohammed, R. Hussain, O. Senko, B. Bimaganbetov, J. Lee, F. Hussain, C. A. Kerrache, E. Barka, and M. Z. Alam Bhuiyan, "A new machine learning-based collaborative ddos mitigation mechanism in software-defined network," in *2018*

14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2018, pp. 1–8.

[92] C. I. for Cybersecurity, "Nsl-kdd dataset." [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[93] Kdd cup 1999 data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[94] Naive bayes classifier. https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c.

[95] Z. Ma and B. Li, "A DDoS attack detection method based on SVM and k-nearest neighbour in SDN environment," *International Journal of Computational Science and Engineering*, vol. 23, no. 3, p. 224, 2020. [Online]. Available: https://doi.org/10.1504/ijcse.2020.111431

[96] trafgen(8) — linux manual page. https://man7.org/linux/man-pages/man8/trafgen.8.html.

[97] Machine learning basics with the k-nearest neighbors algorithm. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[98] 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. 34 699–34 710, 2019.

[99] S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K.-K. R. Choo, and J. Iqbal, "A deep cnn ensemble framework for efficient ddos attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 53 972–53 983, 2020.

[100] W. Liu, X. Liu, X. Di, and H. Qi, "A novel network intrusion detection algorithm based on fast fourier transformation," in *2019 1st International Conference on Industrial Artificial Intelligence (IAI)*, 2019, pp. 1–6.

[101] M. A. Albahar, "Recurrent neural network model based on a new regularization technique for real-time intrusion detection in SDN environments," *Security and Communication Networks*, vol. 2019, pp. 1–9, Nov. 2019. [Online]. Available: https://doi.org/10.1155/2019/8939041

[102] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," in *2020 IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2020, pp. 391–396.

[103] Mininet. http://mininet.org/overview/.

[104] Software-defined nnetworking. https://ryu.readthedocs.io/en/latest/parameters.html.

[105] R. L. S. de Oliveira, C. M. Schweitzer, A. A. Shinoda, and L. R. Prete, "Using mininet for emulation and prototyping software-defined networks," in *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2014.

[106] R. project team, "Ryu sdn framework." [Online]. Available: https://osrg.github.io/ryu-book/en/Ryubook.pdf

[107] A. L. Dean Pemberton and S. Russell, "Ryu openflow controller." [Online]. Available: https://nsrc.org/workshops/2014/nznog-sdn/raw-attachment/wiki/WikiStart/Ryu.pdf

[108] J. Li, "Detection of ddos attacks based on dense neural networks, autoencoders and pearson correlation coefficient," 2020. [Online]. Available: https://dalspace.library.dal.ca/bitstream/handle/10222/78536/Junhong-Li-MCS-FCS-April-2020.pdf?sequence=7

[109] "Pandas library." [Online]. Available: https://pandas.pydata.org/

[110] "Radviz visualizer." [Online]. Available: https://www.scikit-yb.org/en/latest/api/features/radviz.html

[111] "Vgg16 and vgg19." [Online]. Available: https://keras.io/api/applications/vgg/

[112] "Pandas profiling." [Online]. Available: https://pandas-profiling.github.io/pandas-profiling/docs/master/rtd/

[113] M. C. ABOUNAIMA, F. Z. E. MAZOURI, L. LAMRINI, N. NFISSI, N. E. MAKHFI, and M. OUZARF, "The pearson correlation coefficient applied to compare multi-criteria methods: Case the ranking problematic," in *2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, 2020, pp. 1–6.

[114] "The workbench for machine learning." [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/

[115] S. Kiourkoulis, "Ddos datasets," 2020. [Online]. Available: https://ltu.diva-portal.org/smash/get/diva2:1432024/FULLTEXT02.pdf

[116] J. H. Go, T. Jan, M. Mohanty, O. P. Patel, D. Puthal, and M. Prasad, "Visualization approach for malware classification with resnext," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–7.

[117] S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, "Intelligent vision-based malware detection and classification using deep random forest paradigm," *IEEE Access*, vol. 8, pp. 206 303–206 324, 2020.

[118] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware classification with deep convolutional neural networks," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018.

[119] Y. Lu and J. Li, "Generative adversarial network for improving deep learning based malware classification," in *2019 Winter Simulation Conference (WSC)*, 2019, pp. 584–593.

[120] Diabetic retinopathy detection. https://www.kaggle.com/c/diabetic-retinopathy-detection/data.

[121] Chest x-ray images (pneumonia). https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

[122] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, S. Farrukh, and G. A. Shah, "Iot dos and ddos attack detection using resnet," in *2020 IEEE 23rd International Multitopic Conference (INMIC)*, 2020, pp. 1–6.

[123] How to build a convolution neural network based malware detector using malware visualization. https://hub.packtpub.com/how-to-build-a-convolution-neural-network-based-malware-detector-using-malware-visualization-tutorial/.

[124] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[125] "Imagenet." [Online]. Available: https://www.image-net.org/

[126] "Vgg16 – convolutional network for classification and detection." [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/

[127] A. Habibi Lashkari., G. Draper Gil., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, INSTICC. SciTePress, 2017, pp. 253–262.

[128] G. Draper-Gil., A. H. Lashkari., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,* INSTICC. SciTePress, 2016, pp. 407–414.

[129] Openflow controller developer's guide. http://rlenglet.github.io/openfaucet/controller.html.

[130] Evaluating anomaly detection algorithms with precision-recall curves. https://medium.com/wwblog/evaluating-anomaly-detection-algorithms-with-precision-recall-curves-f3eb5b679476.