



UNIVERSIDAD NACIONAL DE COLOMBIA

An Approach for Detection of DDoS Attacks against the Control Plane of Software Defined Networks

Alejandro Alvarez Arguello

Universidad Nacional de Colombia
Facultad de Minas, Departamento de Ciencias de la Computación y la Decisión
Medellín, Colombia
2017

An Approach for Detection of DDoS Attacks against the Control Plane of Software Defined Networks

Alejandro Alvarez Arguello

Thesis submitted as partial requirement for the degree of:
Master in Engineering, Computer Science

Advisor:

Prof. John Willian Branch, PhD (Universidad Nacional de Colombia)

Co-Advisor:

Prof. Sergio Armando Gutiérrez, PhD(c) (Universidad Nacional de Colombia)

Research Line:

Computer Networks

Research Group:

GIDIA

Universidad Nacional de Colombia

Facultad de Minas, Departamento de Ciencias de la Computación y la Decisión

Medellín, Colombia

2017

Abstract

Security of the infrastructure of Software Defined Networks (SDN) is a challenging problem. SDN introduces new threat vectors in addition to those inherited from legacy networks. Thus, it becomes an attractive target for attackers. SDN separates the control and data planes, and migrates the control functions to a logically centralized entity called controller which might be an attractive target for Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks. These attacks can be executed easily using open access tools and without requiring specialized or high performance hardware. According to the literature, the protection of the SDN infrastructure, specially against this kind of threats has not been widely addressed. Thus, we propose an algorithm to detect DDoS attacks against SDN control plane. Our algorithm considers both the OpenFlow traffic towards the control plane and specific interfaces of OpenFlow switches (local perspective detection) or the whole aggregated OpenFlow traffic on the control channel (global perspective detection). In our evaluation, we achieved a 99.94% of accuracy in detecting attacks with a 0.04% of false positives and 0.07% of false negatives.

Keywords: SDN, OpenFlow, DDoS, SPRT, Control plane.

Abstract

La seguridad de la infraestructura de las Redes Definidas por Software (SDN por sus siglas en inglés) es un problema difícil. SDN introduce nuevos vectores de amenaza adicionales a aquellos heredados de las redes tradicionales. SDN se convierte entonces en un objetivo atractivo para los atacantes. SDN separa el plano de control y el plano de datos, y de manera que las funciones de control se migran a una entidad centralizada desde el punto de vista lógico, llamada controlador el cual puede ser un objetivo atractivo para ataques de Denegación de Servicios (DoS) y de Denegación de Servicio Distribuidos (DDoS). Estos ataques pueden ser ejecutados fácilmente usando herramientas de acceso libre y sin requerir hardware especializado o de alto rendimiento. Según la literatura, la protección de la infraestructura SDN, especialmente contra este tipo de amenazas no ha sido abordada ampliamente. Proponemos un algoritmo para detectar ataques DDoS contra el plano de control SDN. Nuestro algoritmo considera el tráfico que pasa entre el plano de control y las interfaces específicas de los suiches OpenFlow (perspectiva local de detección) y todo el tráfico OpenFlow agregado en el canal de control (perspectiva global de de-

tección). En nuestra evaluación, logramos un 99.94% de precisión en la detección de los ataques con un 0.04% de falsos positivos (eventos que no corresponden a ataques) y un 0.07% de falsos negativos (eventos de ataques que fueron ignorados).

Keywords: SDN, OpenFlow, DDoS, SPRT, Plano de control.

Content

1	Introduction	2
1.1	Motivation	2
1.2	Hypothesis and research questions	3
1.3	Proposal	4
1.4	Objectives	4
1.4.1	General	4
1.4.2	Specific	4
1.5	Organization	5
2	Background	6
2.1	Software-Defined Networks (SDN)	6
2.1.1	Basic Architecture	7
2.1.2	OpenFlow	8
2.2	Distributed Denial-of-Service (DoS) attack	11
2.2.1	Classification of DDoS attacks	11
2.2.1.1	Classification by victim type	12
2.2.1.2	Classification by protocol level	12
2.2.2	Anomaly detection and DDoS attacks	13
2.2.2.1	Types of anomaly detection techniques	14
2.3	Threat vectors and vulnerabilities in SDN	15
3	Research problem and relevant techniques	18
3.1	Related-work	18
3.1.1	DDoS control plane attack detection	18
3.1.2	Other approaches	19
3.1.3	Discussion	19
3.1.4	Contribution and Scope	22
3.2	Research problem	22
3.2.1	Problem statement	22
3.2.2	Threat model	23
3.2.3	Attack traffic analysis	24
3.3	Threat vectors and vulnerabilities in the SDN control plane	25
3.4	Analysis of well-known attacks	28
3.5	Analysis of anomaly detection techniques	29

Content

4	Solution description	34
4.1	Detection of DDoS attacks against SDN control plane	34
4.1.1	Detection function based on SPRT	34
4.1.2	Local perspective and global perspective detection functions	36
4.1.3	DDoS detection algorithm	37
5	Evaluation	41
5.1	Analytical evaluation	41
5.1.1	Measures	42
5.1.2	Data sets	42
5.1.3	Parameter settings	44
5.1.4	Experiments	44
5.1.5	Results	45
5.1.5.1	Results using Δ_t^1 and X_{max}^1 on Data Set April 5th	45
5.1.5.2	Results using Δ_t^2 and X_{max}^2 on Data Set April 5th	48
5.1.5.3	Results using Δ_t^3 and X_{max}^3 on Data Set April 5th	48
5.1.5.4	Results using Δ_t^4 and X_{max}^4 on Data Set April 5th	48
5.1.5.5	Results on Data Set April 6th	49
5.2	Implementation	49
5.2.1	Setup	49
5.2.2	Results	51
5.3	Sensibility analysis	51
5.3.1	Varying α and β	52
5.3.2	Varying λ_1 and λ_0	52
5.3.3	Varying Δ_t	54
5.4	Limitations	54
6	Conclusion	55

1 Introduction

Software-Defined Networking (SDN) is a network architecture based on three main principles: (i) separation of control and data planes, (ii) logical centralization of the control operations and unified view of the network state, and (iii) programmability of the network through applications running on the control plane. The combination of these principles and the advantages that they introduce allow to overcome difficulties typically associated to some management tasks on legacy networks such as configuration of complex routing and security policies [Benson et al., 2009].

SDN uses a standardized protocol to program devices, which as anecdotal evidence suggests, it is most widely implemented by OpenFlow-compliant agents in switches [McKeown et al., 2008]. OpenFlow standardizes the communication between the control plane (controller) and the data plane (forwarding devices). The main function of OpenFlow is to program the flow tables of the forwarding devices according to high level network policies defined by network operators [Xia et al., 2015]. Leveraging the principles of the SDN architecture and particularly features of OpenFlow such as the flow-oriented traffic management, novel solutions can be developed in areas such as traffic engineering, wireless and mobility, monitoring, datacenter networks, and security [Kreutz et al., 2015].

1.1 Motivation

A scenario where SDN can provide innovative solutions is network security. The literature reports different use cases such as intrusion detection systems (IDSs) [Shanmugam et al., 2014] [Hu et al., 2013], intrusion prevention systems (IPSs) [Giotis et al., 2014] [Xing et al., 2013], and DDoS attack detection [Zargar and Joshi, 2013] [Li et al., 2014] [Dotcenko et al., 2014] [Braga et al., 2010]. Despite the interest in using SDN to offer solutions in this area, the security of the SDN infrastructure itself has not been widely addressed [Scott-Hayward et al., 2016]. The ability to control the network through software and the logical centralization of the network control introduce new threat vectors and vulnerabilities causing the SDN infrastructure to become an attractive target for attackers [Kreutz et al., 2013] [Benton et al., 2013]. The most critical threat vectors and vulnerabilities are those that enable DDoS attacks against the control plane because these attacks could render unavailable large segments or even the entire network [Yan et al., 2015] [Benton et al., 2013].

1.2 Hypothesis and research questions

DDoS attacks against SDN control plane are difficult to detect using traditional DDoS defense mechanisms because these attacks have some specific features [Dong et al., 2016]. First, it is difficult to detect malicious flows using OpenFlow devices. The OpenFlow devices do not have intelligence to differentiate flows because they only have forwarding functionalities. Second, the traffic of DDoS attacks is directed to the SDN control plane. These attacks have a similar behavior to the one exhibited by reflection-based flooding attacks. These features make the problem DDoS attack detection in the context of SDN hard [Zargar et al., 2013]. In addition, DoS or DDoS attacks against the control plane, specifically against the controller, might be executed by an attacker using easily accessible tools and they do not require specialized or high performance hardware [Kandoi and Antikainen, 2015] [Shin and Gu, 2013].

The literature addresses the problem of detect DDoS attacks against SDN control plane. However, previous proposed methods have some limitations. First, the detection methods could not detect attacks using low rate traffic. Second, detection and mitigation proposals are not capable to detect DDoS attacks based on protocols different to TCP.

1.2 Hypothesis and research questions

In the context of detection of DDoS attacks against SDN control plane, this thesis presents the following hypothesis:

Hypothesis: It is possible to detect DDoS attacks against the SDN control plane through the extraction of OpenFlow traffic features by monitoring the traffic between the controller and OpenFlow switches, in order to detect the attack and if possible, identify the interfaces where the DDoS attacks come from.

In order to guide the investigation conducted in this thesis, the following research questions (RQ) associated with the hypothesis are defined and presented:

RQ1: What are the threat vectors and vulnerabilities of the control plane designs that might be exploited by an attacker to execute DDoS attacks?

RQ2: What is the impact that well-known DoS and DDoS attacks in the context of legacy networks generate when executed in a network environment based on SDN?

RQ3: Which network anomaly detection techniques can be used in the context of SDN to detect attacks aiming at achieving the premises of high accuracy, low false positives rate and minimal overhead during the detection process?

At the end of this study, at least one possible answer for each question is provided. However, this does not mean that different approaches could not accomplish similar results.

1.3 Proposal

During the development of this thesis, we propose an approach to detect DDoS attacks against the control plane of SDN. We leverage the logically centralized control to monitor the OpenFlow traffic looking for abrupt changes in the number of Packet-In messages in comparison to the number of these messages during normal operation of the network.

Our approach tries to identify the sources of a DDoS attack against SDN control plane. To achieve this, we monitor the OpenFlow traffic coming from each interface of OpenFlow devices (local perspective detection). In addition, we complement the per-interface and per-switch analysis through the monitoring of the aggregated OpenFlow traffic of the control communication channel (global detection perspective). Furthermore, if local perspective detection does not detect attacks coming from specific interfaces of the OpenFlow switches (because the attacker uses low rate traffic), global perspective detection tries to detect low rate attacks monitoring the OpenFlow traffic of all OpenFlow switches.

1.4 Objectives

1.4.1 General

Design an algorithm to detect DDoS attacks against the SDN control plane by monitoring the traffic between controller and OpenFlow switches in OpenFlow-based SDN environment, and if possible identify the interfaces of OpenFlow switches where the DDoS attacks come from.

1.4.2 Specific

1. Identify specific threat vectors and vulnerabilities of state-of-art control plane designs that might be target of DDoS attacks.
2. Analyze the impact that well-known DoS and DDoS attacks (reported in the literature in the context of legacy networks) generate in the SDN control plane.
3. Determine a set of network intrusion detection techniques to perform attack detection, applicable in the context of SDN and that aim at achieving the premises of high accuracy, low false positives rate and minimal overhead during the detection process.
4. Specify an algorithm that implements detection of DDoS attacks against SDN control plane by monitoring OpenFlow traffic in switches interfaces (local perspective detection) and the aggregated of OpenFlow traffic in the communication channel (global perspective detection).

1.5 Organization

This thesis is outlined as follows: Chapter 2 presents the fundamentals of our work: Software-Defined networks (SDN), Distributed Denial-of-Service (DDoS) attack, and threat vectors and vulnerabilities in SDN. Chapter 3 shows the related-work, research problem, threat vectors and vulnerabilities in the SDN control plane, analysis of well-known attacks, and analysis of anomaly detection techniques. Chapter 4 describe the DDoS detection algorithm. Chapter 5 present the evaluation and results of the thesis. Finally, Chapter 6 provides a discussion about the results, conclusions and future work.

2 Background

This chapter presents the fundamental concepts used in our work. This chapter is divided into three sections: Software-defined networks (SDN), Distributed Denial-of-Service (DDoS) attacks, threat vectors and vulnerabilities in SDN.

2.1 Software-Defined Networks (SDN)

The main motivation for SDN is to accelerate innovation. In legacy networks, the control and data planes are tightly coupled. As a consequence, the development of new functionality or network features such as new routing algorithms is a very difficult task. These new functionalities imply the modification of the control plane devices through the installation of firmware or even new hardware. The deployment of new functionalities becomes expensive and it is limited by hardware features. In addition, every change in the network topology, configuration or functionality is very complex and tedious [Kreutz et al., 2015] [Benson et al., 2009].

Software-Defined Networking (SDN) is a network architecture where network control is decoupled from forwarding functionality. SDN is based on three main principles [Kreutz et al., 2015]:

1. **Separation of the control and data planes:** Control functionality is removed from the forwarding devices. Thus, they become simple packet forwarding elements with minimal intelligence embedded within them.
2. **Logically centralized control providing a unified view of the topology and network state to applications:** Control logic is migrated to an entity called controller which provides resources and abstractions to facilitate the configuration of the forwarding devices.
3. **Programmability of the network through applications running on the controller:** The network becomes programmable through software applications which can be developed with either general-purpose languages such as Java, Python, and C++ or specific-purpose languages such as Frenetic, Nettle, Netcore, Procera, and Pyretic.

These SDN principles introduce three fundamental abstractions: Forwarding abstraction, distribution abstraction, and specification abstraction. Forwarding abstraction enables the ability to program the forwarding devices, expressing the desired packet forwarding functionality while hiding hardware implementation details. Distribution abstraction changes

2.1 Software-Defined Networks (SDN)

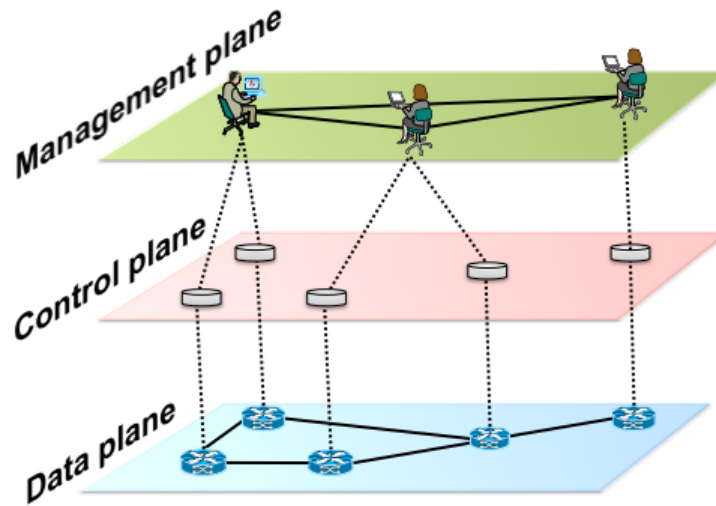


Figure 2-1: Basic Architecture of SDN [Kreutz et al., 2015]

the network distributed control problem to a centralized one. Finally, specification abstraction makes it for network applications to express the desired network behavior without being responsible of their actual implementation in the forwarding devices [Xia et al., 2015].

Compared with legacy networks, SDN architecture has several advantages [Reitblatt et al., 2012]:

1. It is easier to develop network applications due to the offered abstractions (forwarding, distribution, and specification).
2. Network applications can implement consistent and effective network policies leveraging the logically centralized control (control plane).
3. Network applications use the logical centralization of network control and the unified view of the network to dynamically change the configuration of the forwarding devices.
4. The logical centralization of the network control and the unified view of the network state simplify the information sharing among applications. Thus, each application can leverage information provided by other applications to improve its operation.

2.1.1 Basic Architecture

SDN architecture is shown in Figure 2-1. The SDN architecture is composed by the following elements:

1. **Management plane:** The management plane is the set of applications that leverages the functions offered by the northbound interface and the control plane. The man-

2.1 Software-Defined Networks (SDN)

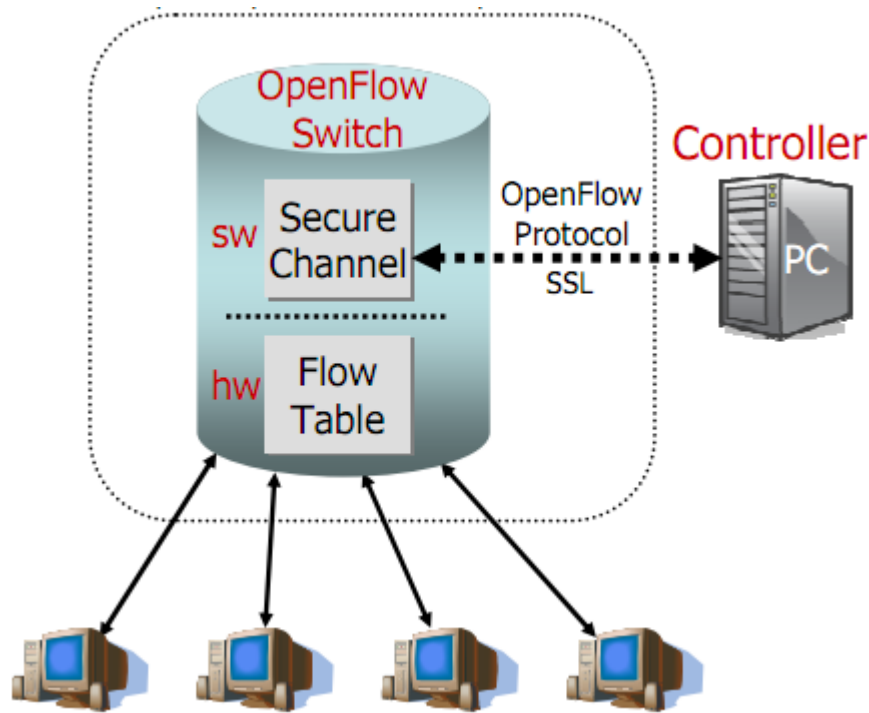


Figure 2-2: Basic architecture of an OpenFlow device [McKeown et al., 2008]

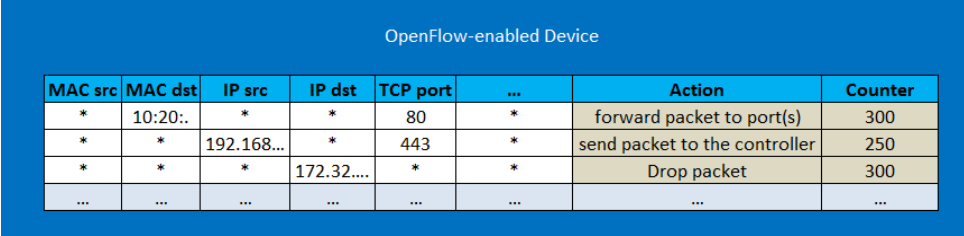
agement plane defines the network policies and remotely monitors and configures the control functionalities using control stations.

2. **Control plane:** The control logic operation of the network resides in the control plane. The control plane translates the high level network policies defined by the network applications and programs the forwarding devices according to these policies. In addition, the control plane provides a unified view of the network state to network applications running on it. The communication between network applications and the control plane is performed through the northbound interface. This interface abstracts to the developers the configuration steps necessary to program the forwarding devices [Kreutz et al., 2015].
3. **Data plane:** Forwarding devices and the connection between them compose the data plane. The data plane elements are programmed by the control plane through the southbound interface. The most common implementation of SDN southbound interface is the OpenFlow protocol [Xia et al., 2015].

2.1.2 OpenFlow

OpenFlow is the protocol used by the control plane to program the forwarding devices. Since OpenFlow has gained significant traction in the industry and academic, with no clear

2.1 Software-Defined Networks (SDN)



The figure shows a table titled "OpenFlow-enabled Device" representing a flow table. The table has columns for matching fields (MAC src, MAC dst, IP src, IP dst, TCP port, ...) and two columns for the result (Action and Counter). The first row has a counter of 300 for the action "forward packet to port(s)". The second row has a counter of 250 for "send packet to the controller". The third row has a counter of 300 for "Drop packet".

MAC src	MAC dst	IP src	IP dst	TCP port	...	Action	Counter
*	10:20:..	*	*	80	*	forward packet to port(s)	300
*	*	192.168...	*	443	*	send packet to the controller	250
*	*	*	172.32....	*	*	Drop packet	300
...

Figure 2-3: Flow table of an OpenFlow device.

competitor as is informally known, we focus on OpenFlow-based SDN architectures in the remainder of this work.

An OpenFlow device is an element that performs packet forwarding according to the configuration defined from the control plane. Figure 2-2 shows the basic architecture of an OpenFlow device.

An OpenFlow forwarding device consists of at least three parts [Xia et al., 2015]:

1. The flow table with matching fields, actions and counters. Figure 2-3 shows an example of a flow table in an OpenFlow device.
2. The bidirectional communication channel with the controller. In the downstream direction (controller to devices), it is used to send messages to program the flow table of the devices. In the upstream direction (devices to controller), it is used by the forwarding devices to request configurations from the controller and reporting statistics or network state changes.
3. The OpenFlow protocol that standardizes the communication between control and data planes.

An OpenFlow device can operate in reactive or proactive mode [Benton et al., 2013]. The reactive mode consists in requesting configuration information to the controller whenever a packet does not match any entry on the flow table. Network applications such as intrusion detection systems (IDS) and network monitoring usually operate in reactive mode [Shirali-Shahreza and Ganjali, 2013]. The reactive mode operation is described as follows [Yan et al., 2015] [Kandoi and Antikainen, 2015]:

1. Whenever a packet is received, the switch looks up for any matching flow rule. A flow rule can be defined as a combination of matching fields.
2. If a match is found, then the action specified in the matching flow rule entry is performed. Also, the statistics (counters) of the entry are updated. In case several rules might match the packet, a priority value assigned to the rule is used to determine which rule to apply. If several rules have the same priority, the first match is applied.

2.1 Software-Defined Networks (SDN)

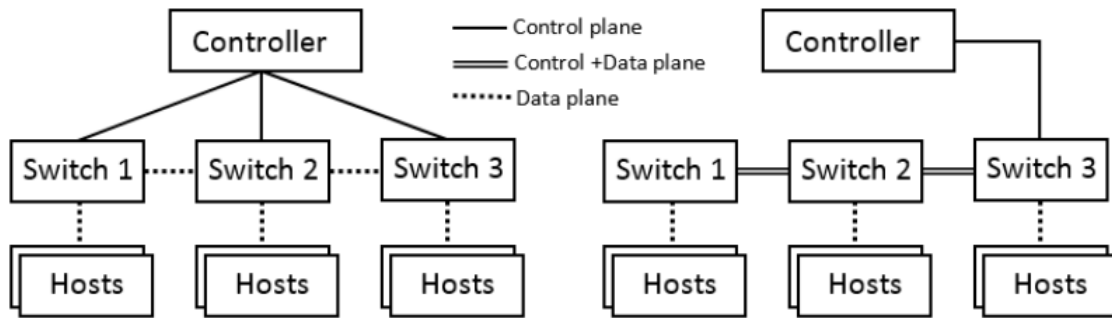


Figure 2-4: Out-of-band (left) and in-band (right) communication design [Kandoi and Antikainen, 2015].

3. If the packet does not match any of the rules in the flow table, then the OpenFlow device generates a Packet-In message. A Packet-In message is an OpenFlow message containing the header or the entire packet. Next, the encapsulated packet is sent to the controller so that it decides what to do with the packet.
4. The controller processes the Packet-In message. When the processing is finished, the controller may discard the packet or may generate a FlowMod message which is sent back to the OpenFlow device. The FlowMod message contains the packet and the configuration information to program the flow table in the OpenFlow device. Thus, the OpenFlow device will not need to generate further Packet-In messages for packets of the same flow, which will match this newly configured flow entry. An exception to this condition would be the case where the flow rule action is actually generating a Packet-In message.

In contrast, the proactive mode consists in programming in advance a set of rules in the OpenFlow devices according to the patterns of traffic that are expected or considered as usual in the network operation.

Finally, the design of the communication channel between the controller and OpenFlow devices can be in-band or out-of-band [Kandoi and Antikainen, 2015]. In in-band design, the communication is performed across the same network infrastructure. It implies that communication between controller and OpenFlow devices must be ensured somehow. In out-of-band communication, there is a dedicated infrastructure for the communication between the controller and the OpenFlow devices. This dedicated infrastructure might consist of separated switching and routing hardware additional to the user network infrastructure. Figure 2-4 shows the out-of-band and in-band communication designs.

2.2 Distributed Denial-of-Service (DoS) attack

Despite many efforts, the magnitude and frequency of DDoS attacks has grown drastically in the last years [Zhang et al., 2016] [Wueest, 2014]. Therefore, they have become a critical threat to the security of different networks.

A Denial of service (DoS) attack tries to disrupt the availability of network resources to legitimate users. If the attack is originated from multiples sources, it is called Distributed Denial of Service (DDoS) attack. DDoS attacks can be launched by two methods [Yan et al., 2015]:

1. By sending malformed or corrupt packets trying to induce crashes, race conditions or unexpected behaviors in software modules implementing protocols or network services. Considering that these modules usually run within the operating system kernel or with privileged permission, the attacks might have catastrophic consequences.
2. By disrupting the connectivity. In these attacks, high volumes of valid packets are sent aiming at exhausting resources of switches, routers and servers such as bandwidth, sockets, CPU and memory.

DDoS attack execution usually follows a DDoS attack strategy with the following typical steps [Zargar et al., 2013]: selection, compromise, communication and attack execution. In the selection step, the attacker identifies hosts that can be incorporated to the attack infrastructure as agents. An agent is a host that has a given vulnerability that can be exploited by the attacker in order to gain access to it. Usually, the user of the host that becomes an agent is not aware of this fact. In the compromise step, the attacker actually exploits the identified vulnerabilities and installs malicious code in the agents. This malicious code is usually composed by backdoors that enable the remote control of the host and the actual programs that perform the DDoS attack. In the communication step, the attacker configures different parameters of the attack code. This configuration is performed via handlers. These are hosts that communicate with the agents using well-known protocols such as TCP, UDP, ICMP or even application protocols such as IRC or HTTP in order to coordinate the agents to launch the attack. Finally, in the execution step, the attack is actually performed. The attacker uses the handlers to trigger the attack execution from the agents.

2.2.1 Classification of DDoS attacks

DDoS attacks have been classified using different criteria. Some exhaustive and complete taxonomies can be found in the literature [Yan et al., 2015] [Zargar et al., 2013] [Peng et al., 2007] [Mirkovic and Reiher, 2004] [Douligeris and Mitrokotsa, 2004]. We review the taxonomies and conclude that the most common classification criteria are victim type and protocol level

2.2 Distributed Denial-of-Service (DoS) attack

2.2.1.1 Classification by victim type

According to the victim type criteria, DDoS attacks can be classified into three categories: application, host, and network [Yan et al., 2015] [Peng et al., 2007].

1. **Application:** They aim at exploiting vulnerabilities in software and/or specific versions of operating systems and even hardware platforms.
2. **Host:** In this case, the target is a particular host. Therefore, all the services running on the host become affected. These attacks usually target network connectivity or in some cases power resources.
3. **Network:** These attacks aim at isolating large segments or even the entire network by targetting elements providing interconnection such as router interfaces or trunk links of switches. They typically require high volumes of traffic.

2.2.1.2 Classification by protocol level

According to the protocol level criteria, DDoS attacks can be classified into two categories: Network/Transport-level and Application-level [Mirkovic and Reiher, 2004].

Network/transport-level

These DDoS attacks are launched using network/transport protocols such as TCP, UDP and ICMP. Network/transport-level attacks can be classified into four sub-categories [Zargar et al., 2013]:

1. **Flooding attacks:** The attackers try to disrupt the connectivity of the legitimate users of the network. The disruption is made by exhausting the bandwidth of the links connected to the victim target sending a high volume of traffic based on TCP, UDP or ICMP protocols. Examples of these attacks are spoofed and non-spoofed UDP flood, ICMP flood, and TCP SYN flood (Neptune) [Peng et al., 2007] [Douligeris and Mitrokotsa, 2004].
2. **Protocol exploitation attacks:** The attackers exploit features and operations of the standard implementation of transport and network protocols which might lead to unexpected situations and/or excessive consumption and exhaustion of CPU or memory resources. Examples of these attacks are TCP SYN-ACK flood, TCP PUSH-ACK flood, and TCP RST/FIN flood [Peng et al., 2007] [Douligeris and Mitrokotsa, 2004].
3. **Reflection/Amplification-based attacks:** These attacks leverage on request/response protocols which are characterized by small requests and large response messages. In these attacks, agents send a request to a server with a spoofed source address. This spoofed address corresponds to the attack target. By performing this action, the responses from the servers (typically larger than the requests) are directed towards the victim (indicated by the spoofed source address). The servers involved in

2.2 Distributed Denial-of-Service (DoS) attack

these attacks are known as reflectors. Examples of these attacks are Smurf, Fraggle, NTP-based attacks, TCP-based attacks and DNS-based attacks [Kührer et al., 2014b] [Peng et al., 2007].

4. **Malformed packet attacks:** The attackers send packets with malformed headers to the victim target trying to confuse some protocol implementations. These packets induce excessive resource consumption in the victim system due to the attempts to parse the malformations in the packets. Some examples of this category are IP fragmentation attack and TCP Kamikaze segments [Douligeris and Mitrokotsa, 2004].

Application-level

Attackers flood with application level protocols the victim server with a few packets that consume a lot of processing resources (CPU, memory and database capabilities) on the victim services. Connectionless protocols such as DNS, SNMP, VoIP and connection-oriented protocols such as FTP, HTTP, SSH and Telnet are used to launch these attacks [Kührer et al., 2014b] [Kührer et al., 2014a]. Examples of these attacks are session flooding attacks, request flooding attacks, asymmetric attacks and repeated one-shot attacks [Ranjan et al., 2009] [Ranjan et al., 2006].

Anomaly detection techniques have been used to detect and mitigate network/transport-level and application-level DDoS attacks in computer networks. Anomaly detection techniques create a model of the normal traffic behavior and then this model is compared with the current network state to detect DDoS attacks [Yan et al., 2015]. Anomaly detection techniques can be used in the context of SDN to protect the control plane.

2.2.2 Anomaly detection and DDoS attacks

Anomaly detection addresses the problem of identifying patterns in data that do not conform to some expected behavior. These non-conforming patterns are commonly called anomalies or outliers [Bhuyan et al., 2014]. There are many application domains for anomaly detection. Some examples are fraud detection, health care, intrusion detection for cyber-security, and fault detection in critical systems. In order to assure the correct operation of systems, detection of anomalies or outliers is important because anomalies or outliers might be symptom of malfunctions or situations that need to be addressed as soon as possible [Chandola et al., 2009]. For example, an anomalous traffic pattern in a computer network could be a symptom of intrusions or attacks.

Network intrusion detection (NID) refers to the application of anomaly detection (AD) in computer networks to find intrusions or attacks [Chandola et al., 2009]. NID refers to detection of malicious activity in computer networks. AD techniques are applicable in NID because the behavior of the network under attack presents noticeable differences with the

2.2 Distributed Denial-of-Service (DoS) attack

behavior in normal conditions [Bhuyan et al., 2014].

NID is based on features extracted from network traffic measurements or system call traces. In order to apply anomaly based network intrusion detection, two things are assumed [Gogoi et al., 2011]:

1. Most of the traffic corresponds to the normal operation of the network and only a small portion of the traffic represents malicious activities.
2. The patterns of the malicious traffic are very different to those that can be observed in normal traffic.

However, in general these assumptions might not be true in real world scenarios. For example, when a network is under an intense DDoS attack, the anomalous traffic is actually more frequent than the normal traffic [Zargar et al., 2013].

2.2.2.1 Types of anomaly detection techniques

NID techniques are classified into three main categories: statistical, machine learning, and data mining techniques.

1. **Statistical:** Statistical techniques fit a statistical model for normal behavior to the given network data and then apply a statistical inference or decision boundary to determine if an unseen instance belongs to the normal traffic. Instances that have a low probability to be generated from the statistical model are considered anomalies. This approach designates an anomaly score to unseen instances and then the score is compared using a statistical test or a predefined threshold to determine whether the unseen instance is anomalous or not [Ahmed et al., 2016] [Patcha and Park, 2007].
2. **Machine learning:** Machine learning techniques improve the ability to distinguish normal behavior from anomalies using a learning process. These techniques construct a function that maps instances to defined classes. To achieve this goal, machine learning mapping functions use labeled training data sets, where each instance in the training data set is labeled with one known class. Machine learning techniques consist of two phases: training and testing. In the training phase, the normal traffic patterns are defined. In the testing phase, the learnt model is applied to new network instances, and every instance in the testing set is classified as normal or anomalous. Machine learning techniques can be divided into classification and clustering techniques [Buczak and Guven, 2015].

The anomaly detection techniques described above can be applied to the context of network intrusion detection. An important feature of anomaly detection techniques is the degree of human intervention in the training and operation on the systems developed using these

2.3 Threat vectors and vulnerabilities in SDN

techniques. In the context of network intrusion detection, the systems developed using anomaly detection are usually classified in supervised, semi-supervised and unsupervised [Chandola et al., 2009].

1. **Supervised:** They require a training process consisting in feeding to the system data sets that indicate if instances in the data sets are normal or intrusions. In the training process, the parameters that control the detection are adjusted by a human operator. NID techniques use training data sets that contain labeled observations for normal and intrusion or attacks classes. Network observations are compared with the normal model to determine if observations belong to normal or intrusion/attack classes.
2. **Semi-Supervised:** They require a partial training process where a data set indicating the normal behavior is fed to the system. They tend to be more applicable but they might be also less accurate. These techniques are more applicable than supervised operation because they do not require labelled intrusions and attacks classes.
3. **Unsupervised:** These systems do not require a previous training. They adjust their control parameters by themselves according to observation of the data and inferences or predictions that are built. These predictions and inferences rely on the assumption that normal behavior is more frequent than abnormal behavior. If this assumption is not true, network intrusion detection techniques might suffer of high false positive rate.

2.3 Threat vectors and vulnerabilities in SDN

The security of SDN by itself has not been widely addressed [Scott-Hayward et al., 2016]. SDN separates the network into three layers: application layer, control layer, and infrastructure layer. Since SDN separates the network into layers, each layer can be vulnerable to DDoS attacks [Kandoi and Antikainen, 2015] [Kreutz et al., 2013].

According to the targets in the SDN architecture, DDoS attacks against SDN are divided into three categories: DDoS attacks against application layer, DDoS attacks against control layer and DDoS attacks against infrastructure layer [Yan et al., 2015].

1. **DDoS attacks against application layer:** The DDoS attacks against application layers are directed to network applications and northbound interfaces. DDoS attacks against network applications and northbound interfaces can affect the normal behavior of the network. For example, a malicious network application can force a behavior on the flow rules in the flow table on forwarding devices to deny all the network traffic.
2. **DDoS attacks against control layer:** Despite the fact that the centralization of the controller is logical rather than physical, it could be seen as a single point of failure. DDoS attacks against control layer can be directed to the controller, northbound

2.3 Threat vectors and vulnerabilities in SDN

Table 2-1: Summarization of previous work in threat vectors and vulnerabilities in SDN.

Authors	Year	Short description
Kandoi and Antikainen	2015	Possible DoS attacks against SDN and their impact
Duner and Kellerer	2015	TLS could be exploited by attackers to execute DDoS attacks against SDN due to the costs of encryption and decryption
Hommes et al.	2014	Implications of DDoS attacks against SDN environment and their impact
Benton et al.	2013	Overview of the vulnerabilities that can be exploited to trigger DoS attacks
Kloti et al.	2013	Security analysis of the OpenFlow protocol
Kreutz et al.	2013	Several threat vectors and vulnerabilities in the SDN/OpenFlow networks
Shin and Gu	2013	Scanner which launches fingerprinting attacks against SDN networks
Fonseca et al.	2012	DDoS attack against SDN based on the generation of packets with random IP sources

interfaces, and southbound interfaces. A successful DDoS attack against control layer, specially against the controller could disrupt the connectivity of the entire network.

- DDoS attacks against infrastructure layer:** DDoS attacks against infrastructure layer are directed towards switches and southbound interfaces. The main idea of these attacks is overwhelming the flow table capabilities of forwarding devices, and/or exhausting the available bandwidth of the control channel. The final effect of this action is hindering or totally disrupting the operation of the forwarding devices.

Some recent works explore the threat vectors and vulnerabilities of SDN. The literature shows that SDN control plane is vulnerable to DDoS attacks. Table 2-1 summarizes the previous work in threat vectors and vulnerabilities in SDN.

Kandoi and Antikainen [Kandoi and Antikainen, 2015] discuss two possible DoS attacks against SDN networks and analyze the impact of these attacks. They show the relationship between timeouts and bandwidth necessary for an attacker to execute a successful DoS attack. They also discuss the necessity of prevention and mitigation strategies to overcome these attacks.

2.3 Threat vectors and vulnerabilities in SDN

Duner and Kellerer [Duner and Kellerer, 2015] explore the current adoption of TLS in the SDN/OpenFlow architecture and present measurements that show the cost of TLS encryption. They study the delay aspects and impact of TLS encryption in controllers and switches. The results show that encryption has a severe impact on control plane performance due to the high resource consumption that it might generate. Therefore, it might be exploited in order to induce DoS attacks.

Hommes et al. [Hommes et al., 2014] show the implications of DDoS attacks against SDN environments and analyze their impact. They conclude that DoS attacks might affect the performance of controller and data plane devices. They propose a DDoS detection method using information theory measurements to analyze the variations in the logical topology.

Benton et al. [Benton et al., 2013] provide an overview of the vulnerabilities that can be exploited to trigger DoS attacks. They show how the vulnerabilities that present DoS risks to SDN. They present that the reactive mode operation could be exploited by attackers to launch a Denial-of-Service (DoS) attacks against SDN platforms.

Kloti et al. [Kloti et al., 2013] present a security analysis of the OpenFlow protocol. They discover vulnerabilities where the attacker sends new flows to generate Packet-In messages directed to the controller. This could lead to Denial-of-Service (DoS) attacks and Information Disclosure. In addition, they elaborate recommendations to prevent and mitigate these vulnerabilities.

Kreutz et al. [Kreutz et al., 2013] present several threats identified in the SDN/OpenFlow networks. They identify seven main threat vectors that affect SDN and propose mechanisms to overcome possible attacks and build a secure and dependable SDN control platform. They show that DoS attacks against the SDN control plane have critical consequences in the SDN infrastructure.

Shin and Gu [Shin and Gu, 2013] introduce a fingerprinting attack against SDN networks. They develop a scanner that generates new flows and measures the delays in order to detect whether a network uses SDN architecture. In addition, they show how this scanner could be used to execute DDoS attacks against SDN architecture.

Fonseca et al. [Fonseca et al., 2012] show a DDoS attack against SDN. This attack is based on the generation of packets with random IP sources from distributed compromised hosts. The high volume of traffic processed by the controller leads to a non-responsive state where the forwarding devices can not communicate with the control plane, which cause the OpenFlow switch not to forward traffic.

3 Research problem and relevant techniques

In this chapter we describe the research problem and relevant techniques considered in the development of the thesis. We present the state-of-the-art, research problem and the development of specific objectives. We divide the chapter into five sections: Related-work, research problem, threat vectors and vulnerabilities in SDN control plane, analysis of well-known attacks, and analysis of anomaly detection techniques.

3.1 Related-work

We make a review of the state-of-the-art looking for DDoS detection methods and other proposals to address the problem of DDoS attacks against the SDN control plane. The literature reports some recent works in detection and mitigation of DDoS attacks against the SDN control plane. We separate the related-work into two categories: DDoS control plane attack detection and other approaches.

3.1.1 DDoS control plane attack detection

Some recent works show DDoS detection methods to detect and mitigate DDoS attacks against the SDN control plane:

Dong et al. [Dong et al., 2016] present a detection method for DDoS attacks against SDN controller. They inject vast number of flows with few packets (low-traffic flows). The detection method is designed to locate the compromised interfaces where malicious attackers are connected. In addition, they show lists of DDoS attacks presented in traditional network that could generate DDoS attacks against SDN controller.

Mousavi et al. [Mousavi and St-Hilaire, 2015] propose an early detection method of DDoS attacks against SDN controllers. The method compares the variations of the entropy of the destination IP addresses of the flows. This measurement determines if the rate of new flows are directed to the same destination.

3.1 Related-work

Ashraf et al. [Ashraf and Latif, 2014] study and suggest the use of machine learning techniques to mitigate intrusion and attacks in SDN with OpenFlow protocol. The proposal presents a comparison of the machine learning techniques which can be applicable to achieve high detection rate with their pros and cons. They conclude that machine learning techniques can be used in the context of intrusion detection.

3.1.2 Other approaches

Some recent work show other approaches used to address the detection and mitigation of DDoS attacks against the SDN control plane.

Mohammadi et al. [Mohammadi et al., 2017] propose a countermeasure to mitigate TCP SYN flooding attacks in SDN. Their proposal takes advantage of dynamic programmability nature of SDN to detect and prevent control plane saturation attacks. However, DDoS attacks against the SDN control plane are possible using protocols different to TCP.

Wang et al. [Wang et al., 2015] propose a protocol-independent defense framework for SDN networks to mitigate data-to-control plane saturation attacks. Their proposal uses proactive flow rule analyzer that derives proactive flow rules by reasoning the runtime logic of the SDN controller and its applications, and a packet migration module that caches the flooding packets and submits them to the controller using rate limit.

Shin et al. [Shin et al., 2013] propose an extension to OpenFlow data plane which reduces the amount of data-to-control plane interactions that arise during control plane saturation attacks. The authors addresses control plane saturation attacks based on TCP protocols. However, SDN control plane is still vulnerable to DDoS attacks based on protocols different to TCP.

3.1.3 Discussion

The literature shows some research works in the discovery of threat vectors and vulnerabilities in SDN environments. DDoS attacks against SDN networks, specially directed to the control plane have severe impact in the performance of the controller and forwarding devices. These attacks are easy to execute and do not need specialized high performance devices.

We find that some DDoS detection methods have been proposed to overcome these issues. Ashraf et al. [Ashraf and Latif, 2014] suggest machine learning techniques to detect and mitigate attacks against SDN infrastructure such as DDoS attacks. However, training and testing process is difficult due to the lack of datasets with labeled attacks against SDN. Mousavi et al. [Mousavi and St-Hilaire, 2015] propose a method for early detection based on

3.1 Related-work

the entropy of the destination IP address. They assume that there exists a difference between the distributions of destination IP under DDoS attacks and normal conditions. However, an attacker can generate a high number of new flows with their destination IP evenly distributed and still overloads the SDN control plane. Finally, Dong et al. [Dong et al., 2016] propose a DDoS detection method based on SPRT to detect a compromised interface during DDoS attacks against the controllers. It assumes that only flows with few packets (low-traffic flows) can overload the SDN control plane. However, their proposed method cannot identify attacks where those attacks are executed using low rate traffic, bypassing the detection and overloading the SDN control plane. In other approaches, the authors propose extensions and frameworks to OpenFlow control and data planes to detect and mitigate DDoS attacks against the SDN control plane. However, the proposals address DDoS attacks based on TCP protocols. Table **3-1** shows an overview of the related-work and their analysis.

However, the related-work addresses the problem with the following limitations:

1. The DDoS detection methods could be bypassed by the attackers using multiples sources using low rate traffic.
2. The proposed mitigation methods detect DDoS attacks based on TCP protocols. An attacker can use protocols different to TCP to execute DDoS attacks against the SDN control plane.

Now, we need to achieve a detection of DDoS attacks against the SDN control plane extracting OpenFlow traffic features in order to detect attacks and if possible, identify the interfaces where the attacks come from. We need to identify specific vulnerabilities in control plane design, analyze the impact of these attacks, determine network intrusion detection techniques applicable in the SDN context to detect the attacks and specify an algorithm for the detection process.

Our DDoS detection algorithm detects DDoS attacks against the SDN control plane based on abrupt changes in the number of Packet-In messages in a defined period of time. This assumption eliminates the limitation of flow-traffic flows made by Dong et al. [Dong et al., 2016] because our detection algorithm processes Packet-In messages coming from all OpenFlow switches to the controller. In addition, our detection algorithm tries to locate the source of the DDoS attack identifying the interfaces on OpenFlow switches that generate abrupt changes in the number of Packet-In messages generated (local perspective detection). If the algorithm does not detect the interfaces of OpenFlow switches due to the low generation rate of Packet-In messages, our algorithm processes the aggregated OpenFlow traffic to detect attacks with low generation traffic rate.

3.1 Related-work

Table **3-1**: Overview of related-work in DDoS attack detection against the SDN control plane.

Authors	Year	Short description	Analysis
Mohammadi et al.	2017	A countermeasure to mitigate TCP SYN flooding attacks in SDN	The solution focusses on detecting TCP SYN flooding attacks. Other attacks are not addressed
Dong et al.	2016	A detection method based on SPRT for DDoS attacks against SDN controllers	The method does not detect attacks based on low rate traffic
Mousavi et al.	2015	An early detection method based on entropy for DDoS attacks against SDN controllers	The entropy measurement could be bypassed using multiple sources with similar traffic rate
Wang et al.	2015	A lightweight and protocol-independent defense framework against data-to-control plane saturation attacks	Data plane cache is still vulnerable to DDoS resource consumption attacks
Ashraf et al.	2014	An study of machine learning techniques to mitigate intrusions and attacks in SDN with OpenFlow protocol	The training of machine learning techniques could be complex in comparison to other techniques
Shin et al.	2013	An extension to the OpenFlow data plane which reduces data-to-control plane interactions	The solution does not address DDoS attacks using protocols different to TCP

3.2 Research problem

3.1.4 Contribution and Scope

Our contribution is a DDoS detection method which:

1. Addresses DDoS attacks using low rate traffic by monitoring the whole aggregated OpenFlow traffic in the control communication channel.
2. Detects the attacks independent to the protocol used by attackers.
3. Tries to identify where the DDoS attacks come from.

The following items delimit the scope of our proposal:

1. We consider OpenFlow as the SDN implementation architecture. Different implementations and realizations of SDN architecture will not be considered.
2. We will address the detection of Network/transport-level DDoS flooding attacks. We do not address application-level or other kinds of DDoS attacks against the SDN control plane.
3. We consider the attacks which target the controllers. Attacks against northbound and southbound interfaces are out of the scope.
4. We consider the attacks which affect controller capabilities. Flow tables capabilities are out of the scope of the thesis.

3.2 Research problem

In this section we present the problem statement, threat model, and attack traffic analysis.

3.2.1 Problem statement

Successful DDoS attacks against the SDN control plane, specially against the controller, could render unavailable a large segment or even the entire network [Yan et al., 2015] [Benton et al., 2013]. These attacks have some specific features that make difficult their detection using traditional DDoS defense mechanisms [Dong et al., 2016]:

1. It is difficult for OpenFlow devices to detect DDoS traffic, specially if the attacker compromises multiple user subnets using low traffic rate attacks.
2. It is difficult to differentiate DDoS traffic and bursty non-malicious traffic.
3. These attacks have a similar behavior of reflection-based flooding attacks, so that the DDoS traffic is not directed to the control plane.

3.2 Research problem

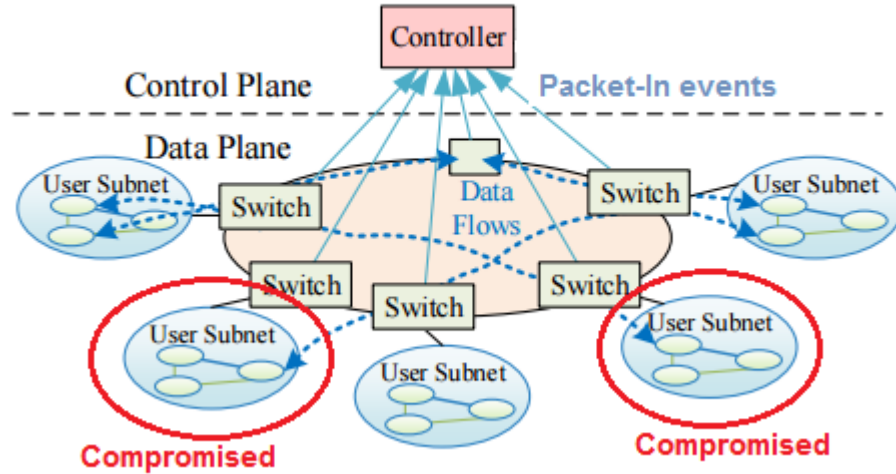


Figure 3-1: Threat model [Dong et al., 2016].

4. DoS and DDoS attacks against the SDN control plane might be executed using easily accessible tools. In addition, these attacks do not require specialized or high performance hardware.

Anomaly detection techniques could be applied in the context of network intrusion to protect the SDN infrastructure against DoS and DDoS attacks [Ashraf and Latif, 2014]. However, the properties of DDoS attacks against the SDN control plane make difficult or even impossible the development of universal detection mechanism to detect all possible variations of these attacks. Furthermore, the selection of anomaly techniques in the context of network intrusion depends on the scenerio where they will be used. Thus, choosing network intrusion detection techniques is not a straightforward task [Yan et al., 2015] [Bhuyan et al., 2014] [Zargar et al., 2013].

3.2.2 Threat model

Figure 3-1 shows the threat model. We employ the same threat model used by Dong. et al. [Dong et al., 2016]. In the basic threat model, the SDN network has a controller, OpenFlow switches and user subnets. OpenFlow switches are connected to the controller. OpenFlow devices are connected between them. We assume that the attacker takes control of one or more hosts inside user subnets. We also assume that the SDN network is working in reactive mode.

During the DDoS attack, we further assume that DDoS attacks can be originated from any

3.2 Research problem

user subnet connected to any OpenFlow switch. In addition, the attacker can generate new flows using the compromised hosts in the user subnets. Considering that the network is operating in reactive mode, when a large number of new flows are created by the attacking hosts, the OpenFlow switches will generate high volume of Packet-In messages towards the controller. Thus, the purpose of the attack is to overload the SDN controller to render it unavailable. This will cause that the OpenFlow switches can not forward traffic coming from the non-compromised hosts in the network.

Dong et al. [Dong et al., 2016] made the assumption that only flows with few packets (low-traffic flows) can be used to launch DDoS attack against SDN control plane. The premise says that the attacker can not execute a effective DDoS attack against SDN control plane if he does not use low-traffic flows. However, it is not difficult to attackers to generate high-traffic flows and damage the SDN control plane. In addition, Dong et al. [Dong et al., 2016] develop their DDoS detection method to find the compromised interface of OpenFlow devices where the attacks comes from. However, it is possible that the attacker executes a DDoS attack using multiple compromised hosts generating a low rate of new flows to overload the controller. We extend the method proposed by Dong et al. [Dong et al., 2016] with the analysis of the aggregated OpenFlow traffic in the communication channel trying to detect DDoS attacks against the SDN control plane when DDoS attacks are generated using low traffic rate.

3.2.3 Attack traffic analysis

The attacker has to discover the conditions that generate new configuration requirements in the flow table in OpenFlow devices (Packet-In events). Then, the attacker sends packets with random headers using the agents to generate new flows matching the new flow generation condition. Next, the OpenFlow switch encapsulates every packet and sends a Packet-In message to the controller. We observe that the features of a DDoS attack against SDN control plane (the controller) are the following:

- The attacker uses the compromised hosts in user subnets to generate new flows.
- Due to the reactive mode operation, OpenFlow devices encapsulate the packet in a Packet-In message and sent it to the controller.
- The controller receives a high volume of Packet-In messages in a short period of time.

We find that the main feature of the attack is the generation of Packet-In messages in a short period of time. Then, we suspect to find DDoS attacks when the network presents abrupt changes in the number of Packet-In message in short periods of time. Figure 3-2 shows an example of the DDoS attack behavior. The red box in the figure represents the moment when network traffic could be a symptom of a suspicious behavior such as a DDoS executed by an attacker.

3.3 Threat vectors and vulnerabilities in the SDN control plane

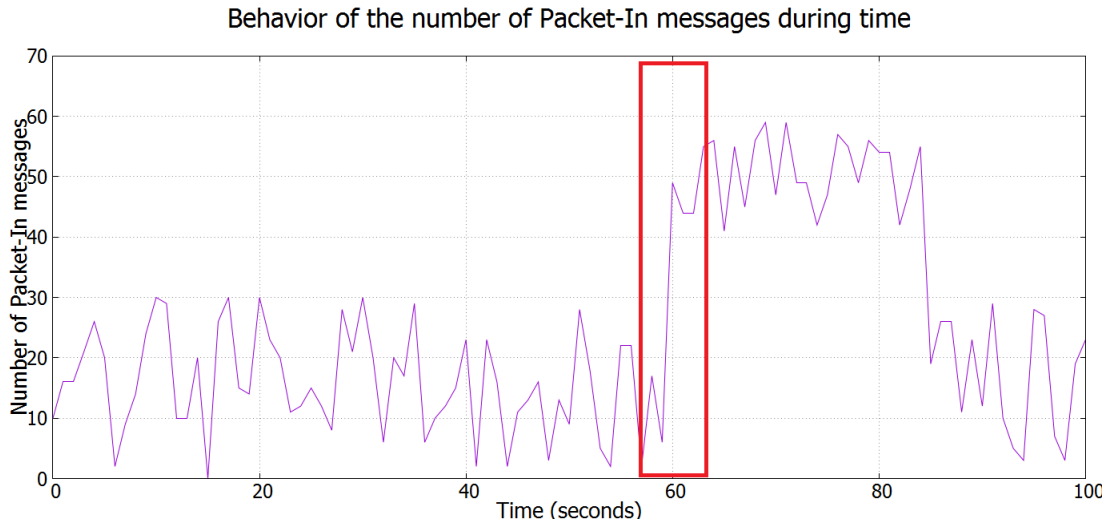


Figure 3-2: Suspicious behavior of the network traffic.

3.3 Threat vectors and vulnerabilities in the SDN control plane

The following corresponds to the development of the specific objective 1. SDN has two properties which makes attractive to attackers. In this section, we analyze the threat vectors and vulnerabilities of the SDN infrastructure. We analyze the vulnerabilities in the control plane that could lead to a successful DDoS attack against SDN control plane.

The Figure 3-3 shows the threat vectors and vulnerabilities of SDN. Kreutz et al. [Kreutz et al., 2013] present seven threat vectors and vulnerabilities of SDN environments: faked traffic flows (threat vector 1), exploitation of vulnerabilities on switches (threat vector 2), attacks against control-data plane communications (threat vector 3), exploitation of vulnerabilities on controllers (threat vector 4), lack of mechanisms to ensure trust between the controller and management applications (threat vector 5), attacks to administrative stations and vulnerabilities presented on them (threat vector 6), and lack of trusted resources for forensics and remediation (threat vector 7).

We find that faked traffic flows (threat vector 1), exploitation of vulnerabilities on switches (threat vector 2), exploitation of vulnerabilities on controllers (threat vector 4) and attacks to administrative stations and vulnerabilities present on them (threat vector 6) might be doors for DDoS attacks against the SDN control plane. These threat vectors are described as follows:

1. **Faked traffic flows (threat vector 1):** Faked traffic flows are flows where packets are created, modified or altered by spoofing one or more headers. The generation of

3.3 Threat vectors and vulnerabilities in the SDN control plane

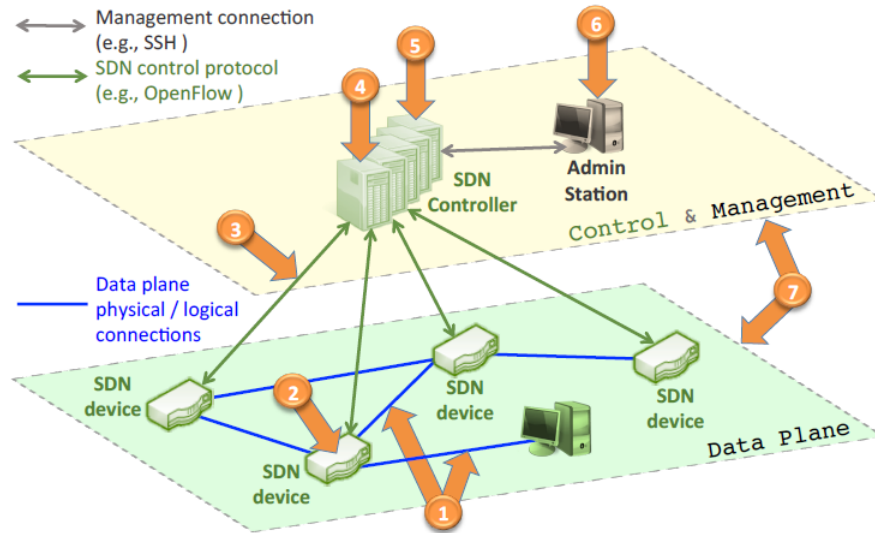


Figure 3-3: SDN threat vector map [Kreutz et al., 2013].

these flows can be used to attack control and data planes targeting the flow tables of OpenFlow switches and controller resources. The attacker could trigger this threat using compromised devices such as switches, servers or personal computers. Authentication mechanisms and TLS communications could help to mitigate the problem where only legitimate users could generate flows. However, if an attacker compromises legitimate users hosts, the threat continues. In addition, authentication mechanisms or TLS communications can generate an additional cost in the SDN infrastructure worsening the impact of the attacks [Durner and Kellerer, 2015].

2. **Exploitation of vulnerabilities on switches (threat vector 2):** An attacker that takes control of OpenFlow switches in SDN infrastructure can use them to drop, clone, inject or slow down packets in the network. If an attacker exploits vulnerabilities in the switches and takes control of them, it is possible to make DDoS attacks against the SDN control plane injecting high volume of Packet-In messages towards the controller.
3. **Exploitation of vulnerabilities on controllers (threat vector 4):** A faulty or malicious controller could compromise the entire network. If an attacker compromises the controller, he can potentially do anything it pleases in the network.
4. **Attacks to administrative stations and vulnerabilities present on them (threat vector 6):** The administrative stations are an exploitable target in the network. If an attacker takes control of administrative stations and these stations control elements such as switches and controllers, he can make malicious actions on the entire network such as DoS and DDoS attacks shutting down the control and data plane elements.

3.3 Threat vectors and vulnerabilities in the SDN control plane

Threat vectors 4 and 6 can be used to shutdown the control plane of the network. However, an attacker can not easily take control of the controller and administrative stations in the SDN infrastructure because these devices are usually isolated or have been heavily hardened due to the importance of these elements in the network. Threat vectors 1 and 2 can be exploited to generate high volume of Packet-In messages directed to the controller. However, if the attacker want to use the threat vector 2 to compromise the controller, he needs to exploit vulnerabilities on switches to take the control of them. Then, the attacker can generate high volume of Packet-In messages and overload the control plane. An attacker can prefer to use the threat vector 1 because taking control of hosts in the network might be easier than compromising forwarding devices. Threat vector 1 permits an attacker to execute easily DDoS attacks against the SDN control plane. In addition, the attacker can generate easily faked packets using the compromised hosts in the network using easily accessible tools.

SDN becomes attractive to attackers due to logical centralization of the control operations in the controller [Kloti et al., 2013]. The controller can be seen as a single point of failure in the network. Despite this notion of single point of failure could be mitigated with physically distributed control plane, each instance might be target of an attack and depending on how the distribution is implemented, the DDoS attacks might still affect the network depending on the operation mode of OpenFlow switches.

An OpenFlow switch can operate in proactive or reactive mode [Benton et al., 2013]. In this mode, the controller has to process the Packet-In messages and then send back to the switch the configuration of flow rules in response to these Packet-In messages. OpenFlow switches which operate in reactive mode might be vulnerable to DDoS attacks [Kandoi and Antikainen, 2015]. If an attacker floods the control plane with faked traffic flows at a high rate and the flooded flows requests arrive at the controller, and they will consume resources such as CPU, memory and bandwidth [Zhang et al., 2016]. If the control plane does not have any protection, the resources of the controller might be exhausted and the controller might crash. When the controller crashes, forwarding devices can not manage new flows and the connectivity becomes unavailable to legitimate users.

Most of DDoS attacks against the SDN control plane have severe impact in networks using reactive mode operation. An attacker can easily overload the control plane if the network operates in reactive mode. Networks using proactive mode operations do not have the same exposure to DDoS attacks unless explicit flow rules force to encapsulate and send Packet-In messages to the controller [Benton et al., 2013].

3.4 Analysis of well-known attacks

The following corresponds to the development of the specific objective 2. We analyze the attacks presented in 1999 DARPA Intrusion Detection Evaluation Data sets [Laboratory, 1999]. We use these data sets because they are widely used in intrusion detection evaluation and contain about 56 types of attacks and 201 attack instances. Throughout our analysis, the packets with the same four-tuple (source IP, source port, destination IP, and destination port), were considered to belong to the same flow. In addition, each packet using network protocol such as ICMP packets (that do not have source IP and destination IP) is considered as an individual flow.

In this section, we analyze how the attacks that are present on the data sets might impact the SDN control plane when they are executed in the context of SDN.

DoS attacks in legacy networks

We analyze the DDoS attacks in the data set and find that the following well-known DDoS attacks can be used for an attacker to execute DDoS attacks against the SDN control plane. The DDoS attacks are: Neptune and Smurf [Dong et al., 2016] [Laboratory, 1999].

1. **Neptune:** This is a SYN flood attack directed to one or more ports in the victim hosts. A SYN flood attack is a form of DoS attack in which an attacker sends high number of SYN requests to a victim host. The attacker attempts to consume server resources to make the system unresponsive to legitimate traffic. Neptune attacks can generate high volume of flows in a short period of time. Neptune attacks can generate high volume of flows in a short period of time, overloading controller.
2. **Smurf:** This is a DDoS attack in which large numbers of ICMP requests with the spoofed source IP of the victim and a broadcast address as a destination IP are sent to a network. Most devices on the network send responses to the source IP address sent in the ICMP request. High number of ICMP responses flood the victim hosts. This attack generates high number of flows in a short period of time.

Probe attacks in legacy networks

We analyze the Probe attacks in the data set and find that the following well-known probes attacks presented in legacy networks can be used for an attacker to execute DDoS attacks against the SDN control plane. The probes attacks are: Portsweep and Ipsweep [Dong et al., 2016] [Laboratory, 1999].

1. **Portsweep:** The attacker scans many ports of a computer system to determine which services are supported by the system. An attacker can use multiples sources to scan many ports of one or more hosts of a network, generating a high number of new flows.

3.5 Analysis of anomaly detection techniques

2. **Ipsweep:** The attacker sends an ICMP request to all hosts of a network to determine which hosts are alive in the network. If the attacker targets a network with many hosts and launches a ipsweep, he can generate high number of flows.

DDoS attacks against the SDN control plane

An attacker could use the generation of faked packets in short periods of time to execute a resource consumption attack of the control plane. If an attacker knows how to generate flows that do not match the flow rules in the flow table of OpenFlow switches, the attacker can use this condition and send many packets with random headers to create new flow rules. These packets will be reported to the control plane as Packet-In messages. The processing of high volume of Packet-In messages consumes a lot of resources and can overload the controller. The attacker can infer the condition that he will use to generate new flow traffic using a SDN fingerprinting attack against SDN environments [Shin and Gu, 2013].

If the attacker knows the condition, he can use it to generate high volume of new flows that are directed to the controller as a Packet-In messages and make the unavailable the connectivity of the user subnets.

In conclusion, SDN attacks against the SDN control plane have severe impact in the SDN infrastructure. The controller can crash due to the high resource consumption produced by high volume of Packet-In messages processed, causing a disruption of the network connectivity as OpenFlow switches can no longer forward traffic. The attacker can employ well-known attacks presented in the legacy networks such as Neptune, Smurf, Portsweep and Ipsweep to execute DDoS attacks against the SDN control plane. These attacks can generate high volume of new flows and can be executed using easily accesible tools.

3.5 Analysis of anomaly detection techniques

The following corresponds to the development of the specific objective 3. We need to choose an anomaly detection technique to be used in our DDoS detection algorithm to detect DDoS attacks against the SDN control plane. We select the anomaly detection techniques aiming at premises of high accuracy, low false positive rate and minimal overhead that literature used in the context of SDN [Dong et al., 2016] [Mousavi and St-Hilaire, 2015] [Ashraf and Latif, 2014] [Xing et al., 2013]. We describe the following anomaly detection techniques: Sequential Probability Ratio Test (SPRT), entropy, Support Vector Machine (SVM) and Rules based.

3.5 Analysis of anomaly detection techniques

Sequential Probability Ratio Test (SPRT)

Sequential Probability Ratio Test (SPRT) is a statistical method to detect anomalies in computer networks. SPRT is a sequential hypothesis that tests a sequential data to determine if the observations belong to normal or anomalous classes. SPRT minimizes the number of successive observations needed to take a decision: accept or reject the null hypothesis.

We consider a simple hypothesis as follows:

$$\begin{cases} H_0 : & \text{normal} \\ H_1 : & \text{anomalous} \end{cases}$$

We can make two types of errors: false positives and false negatives defined as α and β , respectively. We can see α and β as follows:

$$\begin{cases} \alpha = & P(\text{deciding for } H_1 \text{ when } H_0 \text{ is true}) \\ \beta = & P(\text{deciding for } H_0 \text{ when } H_1 \text{ is true}) \end{cases}$$

The next step is calculated the accumulative sum defined as:

$$S_i = S_{i-1} + \log \left(\frac{P(x_i|H_0)}{P(x_i|H_1)} \right)$$

where x_i corresponds to the current observation.

The idea behind the sequential testing gets an observation x_i one at a time and calculates the function S_i to take a decision according to the stopping rule:

$$\begin{cases} A < S_n < B : & \text{collect more observations and repeat the process} \\ S_n \geq B : & \text{accept } H_1 \text{ and stop the hypothesis test} \\ S_n \leq A : & \text{accept } H_0 \text{ and stop the hypothesis test} \end{cases}$$

The values of A and B depend on the values of α and β and should be defined as [Wald, 1973]:

$$\begin{cases} A = & \ln \left(\frac{\beta}{1-\alpha} \right) \\ B = & \ln \left(\frac{1-\beta}{\alpha} \right) \end{cases}$$

Dong et al. [Dong et al., 2016] use SPRT to detect novel DDoS attacks against the SDN control plane. They compare SPRT and entropy and conclude that SPRT can achieve better results according to the measures of accuracy, false positive rate and false negative rate in comparison with entropy solutions.

3.5 Analysis of anomaly detection techniques

Entropy

Entropy is a statistical method to detect anomalies in a traffic data. Entropy measures the randomness in a network. The higher the randomness, the higher is the entropy and vice versa.

Let define W as a set of data x_1, x_2, \dots, x_n where n corresponds to the number of observations of the variable x . The probability of x_i happening in W is:

$$P_i = \frac{x_i}{n}$$

Now, we measure the entropy, referred to as $H(x)$, with the formula:

$$H(x) = - \sum_i^n (P_i * \log(P_i))$$

If all the elements have the same probabilities, the value of the entropy is the maximum. If an element appears more than others, the entropy will be lower. Then, a classification function is defined to normal and anomalous classes as:

$$f(H(x)) = \begin{cases} \text{anomalous}, & H(x) < \text{threshold} \\ \text{normal}, & \text{Otherwise} \end{cases}$$

Mousavi et al. [Mousavi and St-Hilaire, 2015] use entropy of the destination IP addresses to detect DDoS attacks against SDN controllers. If the entropy is low, then a DDoS attack against SDN controller was executed. However, a widely distributed DDoS attack can lead to a maximum entropy and bypass the detection.

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning technique widely used in the context of network intrusion detection. SVM built a model constructing a hyperplane in a high-dimensional space which is used to classify network observations into normal or anomalous classes. The hyperplane separates the normal and anomalous using the largest distance to the nearest training data points between normal and anomalous classes. The distance between the hyper plane that separates the anomaly and normal classes and the closes network data points is maximized and is called the margin. Figure **3-4** shows a simple example of a SVM classifier using a linear hyperplane. Since, linear hyperplane is not applicable to all possible training data, the hyperplane is defined by a kernel function. The

3.5 Analysis of anomaly detection techniques

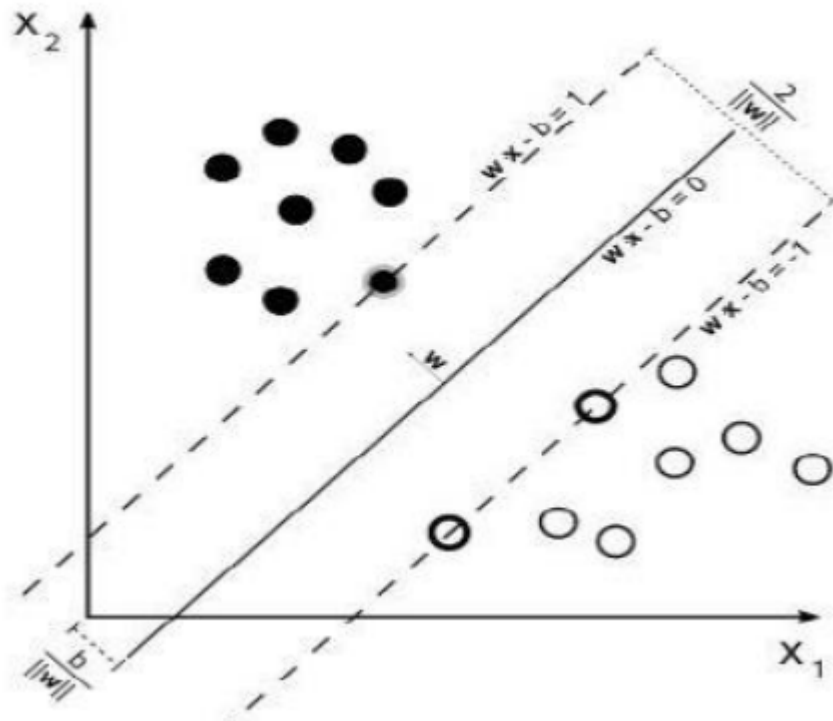


Figure 3-4: Simple example of SVM classifier [Ashraf and Latif, 2014].

kernel function takes the nearest training data points and creates a function that separates the classes. SVM is considering one of the fastest machines learning techniques and usually performs better than other classification techniques.

Ashraf et al. [Ashraf and Latif, 2014] suggest the use of SVM to detect and mitigate attacks against SDN infrastructure because in the context of intrusion detection has shown accuracy results in legacy networks.

SVMs have the advantages that in the context of intrusion detection report attacks with an accuracy of around 98%. However, SVMs need a training dataset with labelled normal and anomalous classes. In addition, the improper distribution of the training network traffic data could lead to low accuracy and high false positive rate.

Rule-based approach

Rule-based approach is one of the most widely used machine learning techniques to detect anomalies in computer networks. Rule-based approach has associated a knowledge base that contains the rules that describe the normal and anomalous classes, and rule engine that matches rules against the current state of the network traffic. Depending on the results of the

3.5 Analysis of anomaly detection techniques

matching, one or more rules can be evaluated and finally reach into normal or anomalous classes. A rule is described as a relationship between different attributes of a network observation. For example, we have an observation with the attributes X and Y. We can define a rule as follows:

IF (X is A) AND (Y is B) THEN (ANOMALOUS TRAFFIC)

Xing et al. [Xing et al., 2013] uses Snort and OpenFlow to detect and mitigate attacks in the network. Snort is a popular open source rule-based IDS that matches each observed packets against a set of rules. Thus, Snort rules identify attacks based on headers such as Ip address, TCP or UDP port numbers, and ICMP headers.

The selected anomaly detection techniques shows high accuracy, low false positive rate, low false positive rate and minimal overhead during the detection process. However, we choose to use of SPRT as our DDoS detection due to comparable results with the others anomaly detection algorithm, but the implementation of the technique is easier than the other techniques. In addition, SPRT has been used in the context of SDN with good results [Dong et al., 2016].

4 Solution description

In this chapter we describe our proposed solution. We present the research problem and the results of each specific objective. We divide the chapter into five sections: Research problem, threat vectors and vulnerabilities in SDN control plane, analysis of well-known attacks, analysis of anomaly detection techniques, and detection of DDoS attacks against SDN control plane.

4.1 Detection of DDoS attacks against SDN control plane

The following corresponds to the development of the specific objective 4. In this section, we present the detection function based on SPRT, we introduce the notion of local perspective and global perspective detection functions, and finally we define our DDoS detection algorithm.

4.1.1 Detection function based on SPRT

In this section, we describe the detection function used in our detection algorithm. The detection function is based on the proposed detection method by Dong et al. [Dong et al., 2016] using SPRT. Let define the variable X_t as the number of Packet-In messages in the interval Δ_t .

We define a function $f(X_t)$ as:

$$f(X_t) = \begin{cases} 1, & \text{if } X_t \geq X_{max} \\ 0, & \text{otherwise} \end{cases} \quad (4-1)$$

Where:

X_{max} is a threshold for the variable X_t

Now, we consider the following hypothesis test:

$$\begin{cases} H_1 : & \text{network under attack} \\ H_0 : & \text{network under normal conditions} \end{cases} \quad (4-2)$$

4.1 Detection of DDoS attacks against SDN control plane

In reality, the detection function can make two types of errors: false positives and false negatives. False positives correspond to the acceptance of H_1 when H_0 is true. False negatives correspond to the acceptance of H_0 when H_1 is true. To avoid these two errors, α and β are defined as the probabilities of false positives and false negatives, respectively. The error rate for false positive and false negative should not exceed α and β .

Next, We define a function S_n $n = 1, 2, \dots, n$ as the evaluation the values f_1, f_2, \dots, f_n . Then, the function S_n evaluates:

$$S_n = \ln \left(\frac{P[f_1, f_2, \dots, f_n | H_1]}{P[f_1, f_2, \dots, f_n | H_0]} \right) \quad (4-3)$$

Each observation of the variable X_t does not depend on the previous observations and has the same distribution. Then, we assume that each value of the function f is independent and identically distributed, and then we have:

$$S_n = \ln \left(\prod_i^n \frac{P[f_i | H_1]}{P[f_i | H_0]} \right) = \sum_i^n \ln \left(\frac{P[f_i | H_1]}{P[f_i | H_0]} \right) \quad (4-4)$$

where:

$$\begin{cases} P[f_i | H_0] \neq \{0, 1\} \\ P[f_i | H_1] \neq \{0, 1\} \end{cases}$$

Let define:

$$P(f(X_t) = 1 | H_1) = 1 - P(f(X_t) = 0 | H_1) = \lambda_1 \quad (4-5)$$

$$P(f(X_t) = 1 | H_0) = 1 - P(f(X_t) = 0 | H_0) = \lambda_0 \quad (4-6)$$

where $\lambda_1 > \lambda_0$ because is more probably that high number of Packet-In messages are injected in a network under attack than under normal conditions. We find a limitation in the values of λ_1 and λ_0 . The detection function works if the values of λ_1 and λ_0 are different to $\{0,1\}$. However, these values of these parameters are unrealistic. If we put a value of $\lambda_1=1$, this means that in a network under attack, the number of flows in a period of time will be always high. If we put a value of $\lambda_1=0$, this means that an attacker can generate DDoS attacks against SDN control plane without the generation of high number of Packet-In messages in a short period of time, attacks such as Portsweep, Smurf and Neptune can generate high number of Packet-In messages in short periods of time. Furthermore, if we put a value of $\lambda_0=1$, this means that a network during normal conditions always has high volume of

4.1 Detection of DDoS attacks against SDN control plane

Packet-In messages. Finally, if we put a value of $\lambda_0=0$, this means that a network during normal conditions never has high volume of Packet-In messages, but bursty nonmalicious traffic could exist.

According to the equations (4-1) (4-3) (4-4) we get:

$$S_n(X_t) = \begin{cases} S_{n-1} + \ln \left(\frac{P(f(X_t)=1|H_1)}{P(f(X_t)=1|H_0)} \right), & \text{if } X_t \geq X_{max} \\ S_{n-1} + \ln \left(\frac{P(f(X_t)=0|H_1)}{P(f(X_t)=0|H_0)} \right), & \text{otherwise} \end{cases} \quad (4-7)$$

According to the equations (4-5) (4-6) (4-7) we get:

$$S_n(X_t) = \begin{cases} S_{n-1} + \ln \frac{\lambda_1}{\lambda_0}, & \text{if } X_t \geq X_{max} \\ S_{n-1} + \ln \frac{1-\lambda_1}{1-\lambda_0}, & \text{otherwise} \end{cases} \quad (4-8)$$

Where $S_0 = 0$.

The values of A and B depend on the values of α and β and should be defined as [Wald, 1973]:

$$\begin{cases} A = \ln \left(\frac{\beta}{1-\alpha} \right) \\ B = \ln \left(\frac{1-\beta}{\alpha} \right) \end{cases}$$

where the value of α and β is chosen between the interval (0, 1). The recommended values of α and β are between 0 and 0.05 [Dong et al., 2016].

Now, we define the detection function based on the equation (4-8) to test whether the network is under attack and under normal conditions. The detection function is a simple thresholding scheme given as follows:

$$D_n = \begin{cases} A < S_n < B : & \text{indicate that need more observations to make a decision (return } -1) \\ S_n \geq B : & \text{accept } H_1 \text{ and reset to the initial values of the detection function (return } 1) \\ S_n \leq A : & \text{accept } H_0 \text{ and reset to the initial values of the detection function (return } 0) \end{cases} \quad (4-9)$$

4.1.2 Local perspective and global perspective detection functions

We separate the functionality of our algorithm in two perspectives: local perspective detection and global perspective detection. Local perspective detection analyzes the OpenFlow

4.1 Detection of DDoS attacks against SDN control plane

traffic passing through specific interfaces in OpenFlow switches. Local perspective detection evaluates the detection function defined above for every interface of OpenFlow switches connected to user subnets. This perspective tries to identify where DDoS attacks come from. Global perspective detection tries to detect DDoS attacks when local perspective detection could not report detection of DDoS attacks. This situation could happen when the attacker launch a DDoS attack using low flow rate. Global perspective evaluates the detection function, taking the number of the aggregated Packet-In messages in the entire control communication channel.

For local detection perspective, we define the variables $X_t^{s,i}$ as the number of Packet-In messages in Δ_t passing through the interface i of the OpenFlow switch s , and $X_{max}^{s,i}$ as the threshold for the function f . For global detection perspective, we define the variable X_t^G as the number of the aggregated Packet-In messages during Δ_t in the control communication channel, and the X_{max}^G as the threshold for the function f . In local perspective detection, we define the functions $f_n^{s,i}$ and $S_n^{s,i}$ as the functions $f(X_t^{s,i}, X_{max}^{s,i})$ and $S_n(X_t^{s,i}, X_{max}^{s,i})$. In global perspective detection, we define the functions f_n^G and S_n^G as the functions $f(X_t^G, X_{max}^G)$ and $S_n(X_t^G, X_{max}^G)$.

Now, the detection function for local perspective detection corresponds to the evaluation of the function D_n using the functions $f_n^{s,i}$ and $S_n^{s,i}$. Global perspective detection evaluates the function D_n using the functions f_n^G and S_n^G .

4.1.3 DDoS detection algorithm

Figure 4.1 shows the DDoS detection algorithm. First, the algorithm reads the parameters $\Delta_t, \lambda_1, \lambda_0, \alpha, \beta, X_{max}^G$ and $X_{max}^{s,i}$. Then, we initialize the variables for local perspective detection and global perspective detection. The variable $status^{s,i}$ and $status^G$ indicate if local perspective detection or global perspective detection is analyzing the variables X_{max}^G and $X_{max}^{s,i}$ and are set true when abrupt changes in the number of Packet-In messages. The variables $alarm^{s,i}$ and $alarm^G$ indicates when the algorithm raises an alarm detection. Then, the algorithm monitors the control communication channel every Δ_t waiting for a Packet-In arrives. If a Packet-In arrives the monitor function is executed. Figure 4.2 shows the monitor function of the DDoS detection algorithm. Next, the algorithm executes the local perspective detection and global perspective detection functions.

Figure 4.3 shows the local perspective detection function. Local perspective detection function evaluates every interface connected to user subnets in every OpenFlow switch in the network. First, the function $f^{s,i}$ is executed to verify an abrupt change in the number of Packet-In messages in the Δ_t . If the function f returns 1, the value of variable $status^{s,i}$ is set true. This means that the local perspective detection is running because an abrupt change

4.1 Detection of DDoS attacks against SDN control plane

Algorithm 4.1 DDoS detection algorithm.

```
1. function DDoSDetectionAlgorithm():
2.   read parameters ( $\Delta_t, \lambda_1, \lambda_0, \alpha, \beta, X_{max}^G, X_{max}^{s,i}$ )
3.   initialize variables
4.     'Variables for Local Perspective Detection and Global Perspective Detection'
5.      $X_t^{s,i}, X_t^G = 0$ 
6.      $status^{s,i}, status^G = false$ 
7.      $S_n^{s,i}, S_n^G = 0$ 
8.      $alarm^{s,i}, alarm^G = false$ 
9.   while (Time) do:
10.    while(inside  $\Delta_t$ ) do:
11.      get Packet-In message
12.      execute Monitor()
13.    endwhile
14.    execute LocalPerspectiveDetection()
15.    if(LocalPerspectiveDetection does not detect an attack):
16.      execute GlobalPerspectiveDetection()
17.    if( $\Delta_{t+1}$ ):
18.       $X_t^{s,i}, X_t^G = 0$ 
19.    endwhile
20. endfunction
```

Algorithm 4.2 Monitor function.

```
1. function Monitor():
2.   'Calculate variables for local perspective detection'
3.   get i = interface
4.   get s = switch
5.    $X_t^{s,i} = X_t^{s,i} + 1$ 
6.   'Calculate variables for global perspective detection'
7.    $X_t^G = X_t^G + 1$ 
8. endfunction
```

4.1 Detection of DDoS attacks against SDN control plane

Algorithm 4.3 LocalPerspectiveDetection function.

```

1. function LocalPerspectiveDetection():
2.   foreach (i = interface, s = OpenFlow switch) do:
3.     if ( $f^{s,i} = 1$ ):
4.        $status^{s,i} = true$ 
5.     endif
6.     if ( $status^{s,i} = true$ ):
7.        $S_n^{s,i} = S_n(X_t^{s,i}, X_{max}^{s,i})$ 
8.        $D_n^{s,i} = D_n(S_n^{s,i})$ 
9.       if ( $D_n^{s,i} = 1$ ):
10.         $alarm^{s,i} = true$ 
11.         $S_n^{s,i} = 0$ 
12.      else if ( $D_n^{s,i} = 0$ ):
13.         $alarm^{s,i} = false$ 
14.         $status^{s,i} = false$ 
15.         $S_n^{s,i} = 0$ 
16.      endif
17.      if ( $alarm^{s,i} = true$ ):
18.        generate alert - Local perspective detection - (Timestamp ,Switch:
s, Interface: i)
19.      endif
20.    endif
21.  endforeach
22. end

```

was found. If the value of the variable $status^{s,i}$ is true, then the algorithm executes the functions $S_n^{s,i}$ and $D_n^{s,i}$. If the value of $D_n^{s,i}$ is 1, then the value of $alarm^{s,i}$ is set to true. This means that an attack is detected and the variable $S_n^{s,i}$ is reseted to 0. By the other hand, if the value of $D_n^{s,i}$ is 0, then the network is under normal conditions and the value of $alarm^{s,i}$ is set to false and the value of $S_n^{s,i}$ is reseted to 0. Finally, the algorithm evaluates the value of the $alarm^{s,i}$. If the value of $alarm^{s,i}$ is true, an attack was detected and the algorithm raise an alarm showing the timestamp, the OpenFlow switch and the interface where the attack comes from. Otherwise, if the value of $alarm^{s,i}$ is false, this means that the network is under normal conditions.

Now, global perspective detection is always applied if local perspective detection could not detect an attack. Figure 4.4 shows the global perspective detection function. Global perspective detection function does the same steps that local perspective detection using the global perspective functions that evaluate the variables X_t^G and X_{max}^G . Finally, the algorithm passes to the next Δ_{t+1} and the values of the variables $X_t^{s,i}$ and X_t^G for local perspective

4.1 Detection of DDoS attacks against SDN control plane

Algorithm 4.4 GlobalPerspectiveDetection function

```
1. function GlobalPerspectiveDetection():
2.   if ( $f^G = 1$ ):
3.      $status^G = true$ 
4.   end
5.   if( $status^G = true$ ):
6.      $S_n^G = S_n(X_t^G, X_{max}^G)$ 
7.      $D_n^G = D_n(S_n^G)$ 
8.     if ( $D_n^G = 1$ ):
9.        $alarm^G = true$ 
10.       $S_n^G = 0$ 
11.     else if ( $D_n^G = 0$ ):
12.        $alarm^G = false$ 
13.        $status^G = false$ 
14.        $S_n^G = 0$ 
15.     end
16.     if ( $alarm^G = true$ ):
17.       generate alert - Global perspective detection - Timestamp
18.     end
19.   end
20. end
```

detection and global perspective detection are set to 0.

5 Evaluation

In this chapter, we present the evaluation of our DDoS detection algorithm. We conduct a series of experiments to answer the following three questions:

1. **What is the accuracy, false positive rate and false negative rate of our DDoS detection algorithm?** Using data sets, we evaluate the measures of accuracy, false positive rate and false negative rate for our DDoS detection algorithm.
2. **How sensitive is our DDoS detection algorithm to parameters settings?** We perform the sensibility analysis to parameter settings. We show how the detection time changes in function of the parameters required.
3. **What are the limitations of our DDoS detection algorithm?** We analyze the results of the accuracy, false positive rate and false negative rate to find the limitation of our DDoS detection algorithm.

We divide the chapter three sections: Analytical evaluation, implementation, sensibility analysis and limitations.

5.1 Analytical evaluation

In this section, we present the analytical evaluation of the DDoS detection algorithm. We evaluate the measures of accuracy, false positive rate and false negative rate of our solution.

The local perspective detection and global perspective detection of the DDoS detection algorithm use the detection function to analyze abrupt changes in the number of Packet-In messages in a period of time. We can evaluate the measures of accuracy, false positive rate and false negative rate evaluating the detection function on either local perspective detection or global perspective detection due to both use the same detection function using different parameters.

In the analytical evaluation, we use data sets to measure the accuracy, false positive rate and false negative rate. Then, we use the detection function on global perspective detection for the analytical evaluation because the data sets do not have interfaces and OpenFlow switches to use local perspective detection.

5.1 Analytical evaluation

5.1.1 Measures

We measure the accuracy, false positive rate and false negative rate for our DDoS detection algorithm using the following metrics:

$$\left\{ \begin{array}{l} \text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \\ \text{False Positive Rate (FPR)} = \frac{FP}{FP+TN} \\ \text{False Negative Rate (FNR)} = \frac{FN}{FN+TP} \end{array} \right.$$

where:

$$\left\{ \begin{array}{l} TP = \text{True positives} \\ TN = \text{True negatives} \\ FP = \text{False positives} \\ FN = \text{False negatives} \end{array} \right.$$

5.1.2 Data sets

We chose the same Data Set used in Dong et al. [Dong et al., 2016] evaluation. We chose the 1999 DARPA Intrusion Detection Evaluation Data Set. This dataset is widely used in intrusion detection and contains both training data without/with attacks and testing data with abundant types of attacks.

The evaluation methodology is described as follows:

1. We use the data sets to calculate an initial parameters setting for the DDoS detection algorithm.
2. We evaluate the initial parameters setting using experiments to determine the best parameters for our DDoS detection algorithm.
3. We evaluate the accuracy, false positive rate and false negative rate for our DDoS detection algorithm.

The data sets used for training and testing are shown as follows:

Data set March 4th: This data set corresponds to the network traffic without attacks. This data set is used to calculate the threshold X_{max} for the detection function on global perspective detection in the DDoS detection algorithm using different values of Δ_t .

5.1 Analytical evaluation

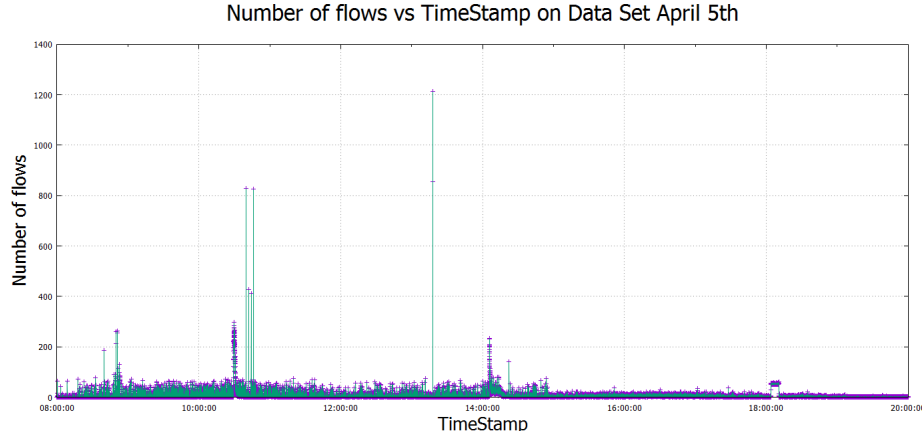


Figure 5-1: Number of flows on Data Set April 5th.

Data set April 5th: This data set is used to determine what values of the parameters Δ_t and X_{max} result in the highest accuracy, lowest false positive rate and lowest false negative rate. The Figure 5-1 shows the number of flows in this data set between 8:00:00 and 20:00:00. Three attacks that might generate high number of flows in a short period of time have been identified in dataset April 5th:

1. **Portsweep:** This attack is executed at 9:43:11 and has duration of 223 seconds.
2. **Smurf:** This attack is executed at 13:18:12 and has duration of 1 second.
3. **Neptune:** This attack is executed at 18:04:04 and has duration of 411 seconds.

The data set shows other points where the number of flows is high. These points are considered as bursty non-malicious flows. We evaluate the metrics of accuracy, false positive rate and false negative rate considering and excluding the Portsweep attacks due to these attacks could generate or not high volume of new flows.

Data set April 6th: This corresponds to a testing data set. It is used for the testing process. The Figure 5-2 shows the number of flows between 8:00:00 and 20:00:00. Two attacks that might generate high number of flows in a short period of time have been identified in data set April 6th:

1. **Neptune:** This attack is executed at 11:38:04 and has duration of 821 seconds.
2. **Neptune:** This attack is executed at 18:16:05 and has duration of 206 seconds.

5.1 Analytical evaluation

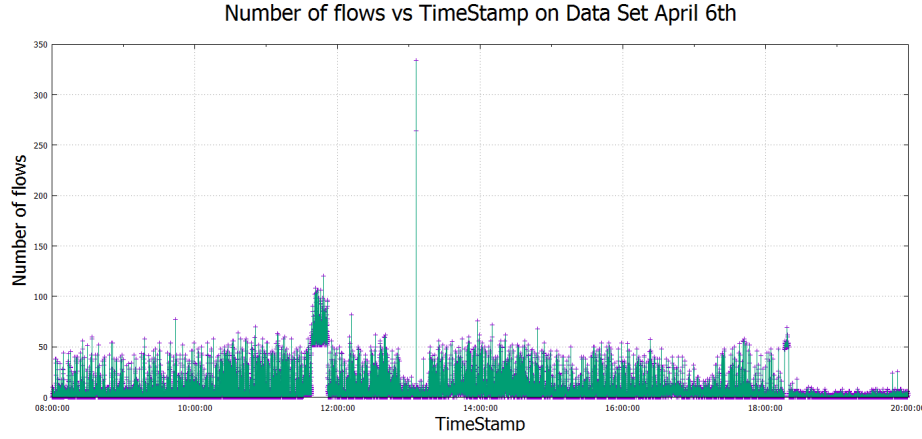


Figure 5-2: Number of flows on Data Set April 6th.

5.1.3 Parameter settings

Table 5-1 shows the values of parameter settings that we use in the analytical evaluation. The values of the parameters of the DDoS detection algorithm are chosen as follows:

1. Δ_t : We define the domain for Δ_t as the list of values 0.1, 0.2, 0.5 and 1.0 seconds.
2. X_{max} : We choose the value of X_{max} as the mean of the number of flows plus three times the value of the standard deviation for each Δ_t . We calculate the value of X_{max} for each Δ_t using the dataset March 4th and count the number of flows in the Δ_t that are normal according the X_{max} . We find that 96% of the number of flows in Δ_t are considered as normal analyzing the data set March 4th.
3. $\alpha, \beta, \lambda_1, \lambda_0$: We use the same parameter settings used by Dong et al. [Dong et al., 2016]. The values of $\alpha, \beta, \lambda_1, \lambda_0$ are 0.01, 0.02, 0.6 and 0.33, respectively.

5.1.4 Experiments

The experiments for the analytical evaluation are the following:

1. Run the algorithm using Δ_t^1 and X_{max}^1 on Data Set April 5th.
2. Run the algorithm using Δ_t^2 and X_{max}^2 on Data Set April 5th.
3. Run the algorithm using Δ_t^3 and X_{max}^3 on Data Set April 5th.
4. Run the algorithm using Δ_t^4 and X_{max}^4 on Data Set April 5th.
5. Run the algorithm on Data Set April 6th.

5.1 Analytical evaluation

Table 5-1: Parameters settings values.

Parameter	Value
Δ_t^1	0.1
Δ_t^2	0.2
Δ_t^3	0.5
Δ_t^4	1.0
X_{max}^1	8.0
X_{max}^2	11.0
X_{max}^3	17.0
X_{max}^4	26.0
α	0.01
β	0.02
λ_1	0.6
λ_0	0.33

We divide the experiments in training and testing experiments. The first four experiments corresponds to the training experiments. The training experiments are done to compare the results between them and select the parameters that result with the highest accuracy, lowest false positive rate and lowest false negative rate. Finally, the last experiment corresponds to the testing experiment. The testing experiment evaluates the algorithm using a testing data set using the metrics of accuracy, false positive rate and false negative rate.

5.1.5 Results

We show the results of the training and testing experiments. The Figures 5-3, 5-4, 5-5, 5-6 and 5-7 shows the results of the training and testing experiments. The Table 5-8 and Table 5-9 shows the metrics of the training experiments considering and excluding Portsweep attacks, respectively. The Table 5-10 shows the metrics for the testing experiments.

5.1.5.1 Results using Δ_t^1 and X_{max}^1 on Data Set April 5th

For this experiment we run the DDoS detection algorithm using the parameters Δ_t^1 and X_{max}^1 . If we consider the Portsweep attack, we achieved a 98.12% of accuracy with 0.77% of false positives and 72.16% of false negatives. If we do not consider the Portsweep attack, we achieve a 98.72% of accuracy, 0.77% of false positives and 95.67% of false negatives. The algorithm does not detect all the traffic generated by the Portsweep. The traffic generated by the Smurf attack is detected. However, the most of the attack traffic is generated by the Neptune attack. Only some traffic generated by the Neptune attack is detected by the algorithm. These situation leads to an extremely high false negative rate.

5.1 Analytical evaluation

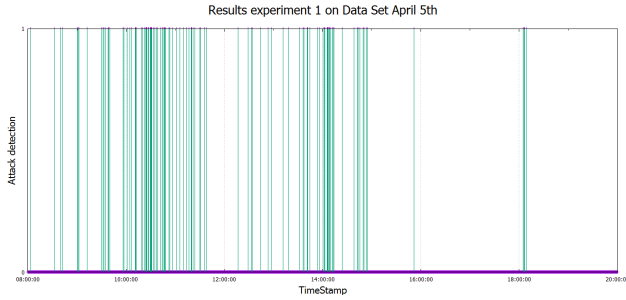


Figure 5-3: Results using Δ_t^1 and X_{max}^1 on Data Set April 5th.

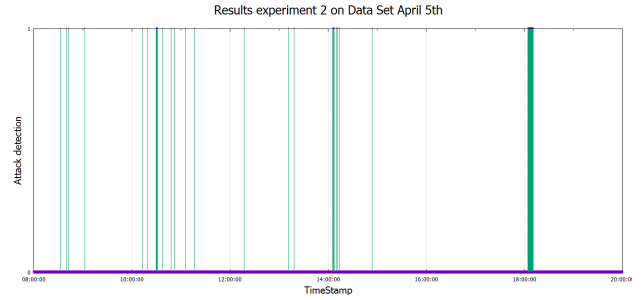


Figure 5-4: Results using Δ_t^2 and X_{max}^2 on Data Set April 5th.

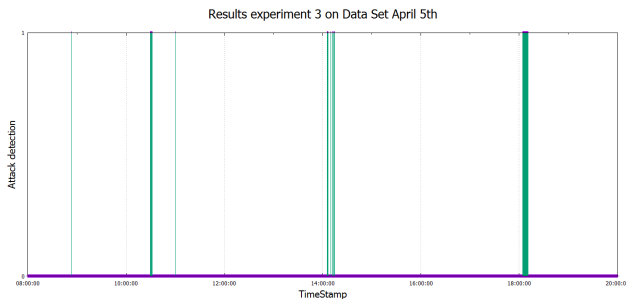


Figure 5-5: Results using Δ_t^3 and X_{max}^3 on Data Set April 5th.

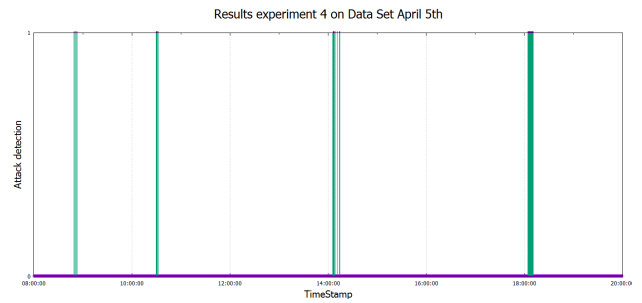


Figure 5-6: Results using Δ_t^4 and X_{max}^4 on Data Set April 5th.

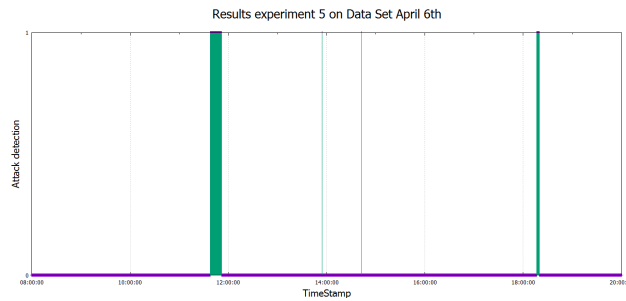


Figure 5-7: Results using the best Δ_t and X_{max} on Data Set April 6th.

5.1 Analytical evaluation

Figure 5-8: Metrics of the training experiments considering Portsweep.

Experiment	Accuracy	FPR	FNR
Results using Δ_t^1 and X_{max}^1	98.12%	0.77%	72.16%
Results using Δ_t^2 and X_{max}^2	98.86%	0.52%	44.97%
Results using Δ_t^3 and X_{max}^3	98.68%	0.85%	17.22%
Results using Δ_t^4 and X_{max}^4	98.37%	1.17%	18.15%

Figure 5-9: Metrics of the training experiments without considering Portsweep.

Experiment	Accuracy	FPR	FNR
Results using Δ_t^1 and X_{max}^1	98.72%	0.77%	95.67%
Results using Δ_t^2 and X_{max}^2	99.12%	0.52%	41.52%
Results using Δ_t^3 and X_{max}^3	99.23%	0.77%	0.49%
Results using Δ_t^4 and X_{max}^4	98.85%	0.001%	0.01%

Figure 5-10: Metrics of testing experiments.

Metric	Values
Accuracy	99.94%
FPR	0.04%
FNR	0.07%

5.1 Analytical evaluation

5.1.5.2 Results using Δ_t^2 and X_{max}^2 on Data Set April 5th

For this experiment we run the DDoS detection algorithm using the parameters Δ_t^2 and X_{max}^2 . If we consider the Portsweep attack, we achieved a 98.86% of accuracy with 0.52% of false positives and 44.97% of false negatives. If we do not consider the Portsweep attack, we achieve a 99.12% of accuracy, 0.52% of false positives and 41.52% of false negatives. This experiment shows results similar to the parameters delta1 and delta1. Only some traffic of the Portsweep and Neptune attack is considered an attack. The traffic generated by the smurf attack is detected by the algorithm. The results of the experiments for Δ_t^1, X_{max}^1 and Δ_t^2, X_{max}^2 leads to excessive false negative rate. These parameters will not be used in the testing experiment.

5.1.5.3 Results using Δ_t^3 and X_{max}^3 on Data Set April 5th

For this experiment, we run the DDoS detection algorithm using the parameters Δ_t^3 and X_{max}^3 . If we consider the Portsweep attack, we achieved a 98.68% of accuracy with 0.85% of false positives and 17.22% of false negatives. The value of the false positive rate is high because the Portsweep attack is not detected by the algorithm because this attack generates low number of flows in a short period of time. If we do not consider the Portsweep attack, we achieve a 99.23% of accuracy, 0.77% of false positives and 0.49% of false negatives. The traffic generated by the Smurf attack is not detected in both cases. This situation happens due to the duration of the attack (1 second). The traffic generated by the Neptune attack is detected and the algorithm raises detection.

5.1.5.4 Results using Δ_t^4 and X_{max}^4 on Data Set April 5th

For this experiment, we run the DDoS detection algorithm using the parameters Δ_t^4 and X_{max}^4 . If we consider the Portsweep attack, we achieved a 98.37% of accuracy with 1.17% of false positives and 18.15% of false negatives. We analyze the false negative rate and we find that this value is due to the algorithm does not detect the Portsweep attack. If we do not consider the Portsweep attack, we achieve a 98.85% of accuracy, 0.001% of false positives and 0.01% of false negatives. In addition, the algorithm does not detect the traffic generated by Smuft attack because the short duration of the attack (1 second). In both cases, the traffic generated by the Neptune attack is detected.

We find that the parameters that show the highest accuracy, lowest false positive rate and lowest false negative rate correspond to Δ_t^3 and X_{max}^3 . The parameters Δ_t^3, X_{max}^3 and Δ_t^4, X_{max}^4 show similar results. However, the parameters Δ_t^3 and X_{max}^3 reduce the time of the detection. These parameters are used in the testing experiment using the dataset April 6th.

5.2 Implementation

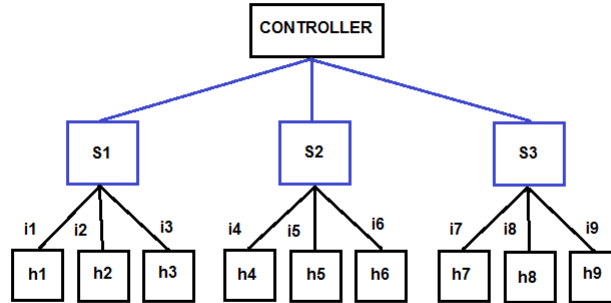


Figure 5-11: Mininet topology used in the implementation.

5.1.5.5 Results on Data Set April 6th

For this experiment we run the DDoS detection algorithm using the parameters Δ_t and X_{max} that shows the highest accuracy, lowest false positive rate and lowest false negative rate. These parameters correspond to the values of Δ_t^3 and X_{max}^3 . In this experiment, we achieved an accuracy of 99.94% with a 0.04% of false positives and 0.07% of false negatives. In addition, we found that the algorithm has a delay to raise a detection of the two attacks of four seconds.

5.2 Implementation

We have implemented our DDoS detection algorithm. The implementation evaluates how the DDoS detection algorithm works in real environments. We show the setup and the results.

5.2.1 Setup

We program the DDoS detection algorithm that consists in an script in Python. Table 5-1 shows the parameter setting used in the implementation based on the results of the analytical evaluation. We simulate a simple topology using Mininet. Figure 5-11 shows the topology. The topology consists in one controller, and three OpenFlow switches connected to it. We connect four hosts to every OpenFlow switch simulating the user subnets. In addition, we use the POX controller installed in the Mininet virtual machine as the SDN controller. The flows are managed using the `l2_learning` program of POX controller and the attacks are launched using a script in Python that we program that generates packets using random headers to create new flows. The normal traffic is simulated using the ping command of the hosts.

5.2 Implementation

Table 5-2: Parameters settings values for the implementation.

Parameter	Value
Δ_t^1	0.5
$X_{max}^{s,i}$	26.0
X_{max}^G	26.0
α	0.01
β	0.02
λ_1	0.6
λ_0	0.33

We monitor the OpenFlow traffic during 510 seconds and execute the following DDoS attacks:

1. **Attack 1:** We evaluate the local perspective detection. We execute the script of the attack during 1 second in the 30 seconds of the interval using the hosts 1 and 2.
2. **Attack 2:** We evaluate the local perspective detection. We execute the script of the attack during 2 seconds in the 90 seconds of the interval using the hosts 1 and 4.
3. **Attack 3:** We evaluate the local perspective detection. We execute the script of the attack during 4 seconds in the 150 seconds of the interval using the hosts 1, 4 and 7.
4. **Attack 4:** We evaluate the local perspective detection. We execute the script of the attack during 10 seconds in the 210 seconds of the interval using the hosts 2, 5 and 8.
5. **Attack 5:** We evaluate the local perspective detection. We execute the script of the attack during 20 seconds in the 270 seconds of the interval using the hosts 6.
6. **Attack 6:** We evaluate the local perspective detection. We execute the script of the attack during 30 seconds in the 330 seconds of the interval using the hosts 1.
7. **Attack 7:** We evaluate the global perspective detection. We execute an attack using the pingall command in Mininet to simulate a IpSweep attack inside the network. We execute this attack during 15 seconds in the 390 seconds in the interval using all the hosts.
8. **Attack 8:** We evaluate the global perspective detection. We modify the script of the attack to send low rate of forged packets and execute the script during 40 seconds in the 450 seconds of the interval using all hosts.

The DDoS attacks are executed using different hosts connected to a different OpenFlow switches to evaluate the behavior of the algorithm coming from different OpenFlow switches.

5.3 Sensibility analysis

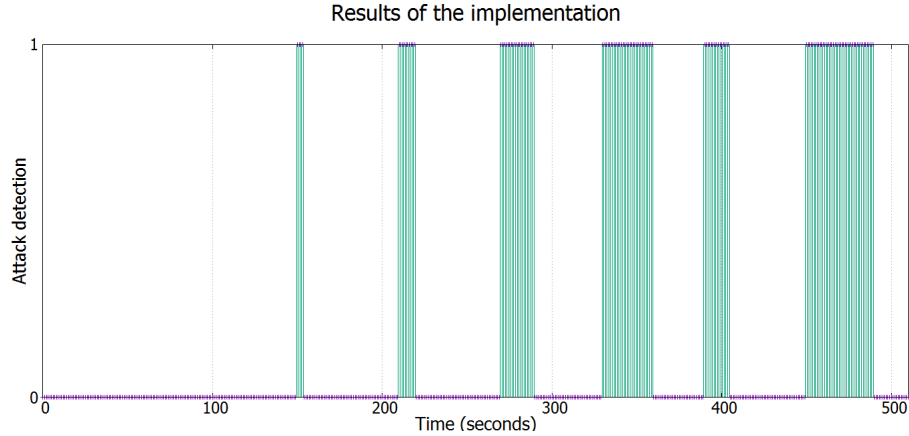


Figure 5-12: Results of the implementation.

5.2.2 Results

In this experiment, we achieved an accuracy of 99.62% with a 0.06% of false positives and 1.2% of false negatives detecting the eight attacks. Figure 5-12 shows the points where the DDoS detection algorithm raise an alarm. Table 5-3 shows the individual results of the implementation experiments. We analyze each attack to discover features of our DDoS detection algorithm in a real environment. The attacks 1 to 6 evaluate the local perspective detection. The DDoS detection algorithm does not detect the attacks 1 and 2. These attacks are reflected in the false positive rate. The algorithm detects the attack 3 using the local perspective detection and reports the hosts 1, 4 and 7 as the source of the DDoS traffic. The attack 4 is detected and the algorithm reports that hosts 2, 5 and 8 are the sources of the DDoS traffic. The attacks 5 and 6 are detected and the algorithm reports the hosts 6 and 1 as the source of the DDoS traffic for the attack 5 and 6, respectively. The attack 7 and 8 evaluates the global perspective detection of the DDoS detection algorithm. The algorithm detects these two attacks but they are shown by the global perspective detection because the source of the attacks are not detected by local perspective algorithm.

5.3 Sensibility analysis

We perform a sensibility analysis on the DDoS detection algorithm. The purpose of this analysis is show the variability of the number of successive observations needed by the algorithm to detect an attack. The number of successive observations needed to detect a DDoS attack against SDN control plane affects the detection time of the algorithm because each observation is performed in a Δ_t .

We perform the sensibility analysis as follows:

5.3 Sensibility analysis

Table 5-3: Individual results of the implementation experiments.

Experiment	Perspective	Detection	Observation
Attack 1	Local	Not detected	Attack is not detected due to the duration of the attack
Attack 2	Local	Not detected	Attack is not detected due to the duration of the attack
Attack 3	Local	Detected	Hosts 1, 4 and 7 are reported
Attack 4	Local	Detected	Hosts 2, 5 and 8 are reported
Attack 5	Local	Detected	Host 1 is reported
Attack 6	Local	Detected	Host 6 is reported
Attack 7	Global	Detected	The source of the attack is not reported
Attack 8	Global	Detected	The source of the attack is not reported

5.3.1 Varying α and β .

We fix the values of λ_1 and λ_0 as 0.6 and 0.33, respectively. These values are chosen according to the parameters setting defined in the Table 5-1. Then, we vary the values of α and β between 0.05 and 0.5. The Figure 5-13 shows the variability of the number of successive observations needed by the algorithm to detect an attack. The number of successive observations increases when α and β are closer to zero. The Figure 5-13 shows that the maximum value the number of successive observations between 0.05 and 0.5 is 11 observations. This means that the maximum time needed to detect a DDoS attack against SDN control plane is around 11 times the value of Δ_t .

5.3.2 Varying λ_1 and λ_0 .

We fix the values of α and β as 0.01 and 0.02, respectively. These values are chosen according to the parameters setting defined in the Table 5-1. Then, we vary the value of λ_1 between 0.5 and 0.7 and the value of λ_0 between 0.4 and 0.5. We choose these intervals because the DDoS detection algorithm consider that $\lambda_1 > \lambda_0$. The Figure 5-14 shows the variability of the number of successive observations needed by the algorithm to detect an attack. We find that the number of successive observations increases when the values of λ_1 and λ_0 are closer between them and viceversa.

5.3 Sensibility analysis

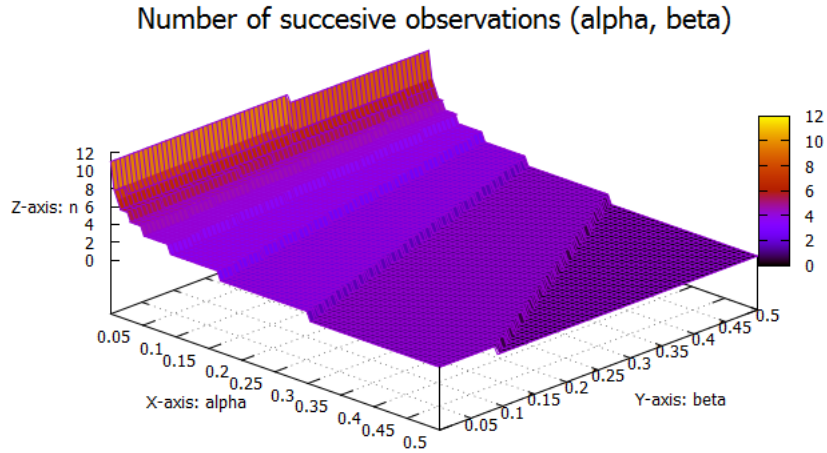


Figure 5-13: Number of successive observations depending of the parameters α and β .

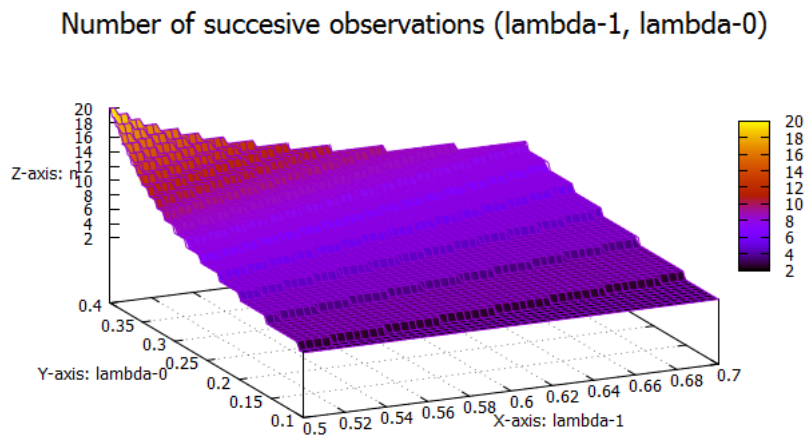


Figure 5-14: Number of successive observations depending of the parameters λ_1 and λ_0 .

5.4 Limitations

5.3.3 Varying Δ_t .

We analyze the time needed by the algorithm to detect an attack varying the value of Δ_t . We fix the values of α, β, λ_1 and λ_0 as 0.01, 0.02, 0.6 and 0.33, respectively. These values are chosen according to the defined parameters in the Table 5-1. According to the Figure 5-13 and Figure 5-14, the DDoS detection algorithm using these parameters needs at least 7 successive observations to detect an DDoS attack against the SDN control plane. Each successive observations is made in Δ_t . We conclude that our DDoS detection algorithm is highly dependent of the parameter Δ_t because this value defines the detection time of the algorithm. For example, we define the value of Δ_t as 0.5 seconds in our analytical evaluation and this means that the algorithm need at least 3.5 seconds to detect an attack taking 7 successive observations of Δ_t .

5.4 Limitations

During the evaluation, we find some limitations presented in our DDoS detection algorithm. First, the algorithm can not to detect attacks with a short time duration such as Smurf which has a duration of one second. Second, the algorithm can not detect attacks with a duration under the 3.5 seconds. The algorithm needs at least 7 successive observations to detect an attack and this results in at least 3.5 seconds analyzing the successive observations. If the attacker executes attacks with a duration under the 3.5 seconds such as Smurf, the algorithm does not detect the attack and the controller can be overloaded.

6 Conclusion

SDN can provide innovative solutions in network security. However, the security of SDN architecture has not been widely addressed. SDN architectures that operate in reactive mode could be very vulnerable to DDoS attacks against the SDN control plane. An attacker can execute a DDoS attack against the SDN control plane by generating high number of new flows. These new flows generate a high volume of Packet-In messages to the controller causing a crash of the controller due to high resource consumption.

We propose the design of an algorithm to detect DDoS attacks against the SDN control plane. We leverage the logical centralized control to monitor the OpenFlow traffic of the interfaces of the OpenFlow switches and the aggregated OpenFlow traffic of the SDN communication channel. We make the assumption that DDoS attacks against SDN control plane shows an abrupt changes in the number of Packet-In messages during the time. Our algorithm considers both the OpenFlow traffic coming from a specific interface of an OpenFlow switch towards the control plane (local perspective detection) and the whole aggregated OpenFlow traffic on the control channel (global perspective detection). In our evaluation, we achieved a 99.94% of accuracy in detecting DDoS attacks against SDN control plane with a 0.04% of false positives and 0.07% of false negatives. However, we identify some additional issues that need to be improved. Our DDoS detection algorithm needs at least four seconds to raise detection. An attacker can generate enough high volume of new flows during the first four seconds of the attacks and crash the controller. In addition, attacks such as Portsweep could not be detected if the attacker generates low number of flows and generate high false negative rate.

For our future work, we will improve the false negative rate and the detection time of our DDoS detection algorithm. We will explore the use of misuse detection techniques to create signatures for Portsweep, Smurf and Neptune to make a quick detection of these attacks that could be used to generate attacks against the SDN control plane.

Bibliography

- [Ahmed et al., 2016] Ahmed, M., Mahmood, A. N., and Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31.
- [Ashraf and Latif, 2014] Ashraf, J. and Latif, S. (2014). Handling intrusion and ddos attacks in software defined networks using machine learning techniques. In *Software Engineering Conference (NSEC), 2014 National*, pages 55–60. IEEE.
- [Benson et al., 2009] Benson, T., Akella, A., and Maltz, D. (2009). Unraveling the Complexity of Network Management. *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, pages 335–348.
- [Benton et al., 2013] Benton, K., Camp, L. J., and Small, C. (2013). OpenFlow vulnerability assessment. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, page 151.
- [Bhuyan et al., 2014] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2014). Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials*, 16(1):303–336.
- [Braga et al., 2010] Braga, R., Mota, E., and Passito, A. (2010). Lightweight DDoS flooding attack detection using NOX/OpenFlow. *Proceedings - Conference on Local Computer Networks, LCN*, pages 408–415.
- [Buczak and Guven, 2015] Buczak, A. and Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, PP(99):1.
- [Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15.
- [Dong et al., 2016] Dong, P., Du, X., Zhang, H., and Xu, T. (2016). A detection method for a novel ddos attack against sdn controllers by vast new low-traffic flows. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–6. IEEE.
- [Dotcenko et al., 2014] Dotcenko, S., Vladyko, A., and Letenko, I. (2014). A Fuzzy Logic-Based Information Security Management for Software-Defined Networks. *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, pages 167–171.

Bibliography

- [Douligeris and Mitrokotsa, 2004] Douligeris, C. and Mitrokotsa, A. (2004). Ddos attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks*, 44(5):643–666.
- [Durner and Kellerer, 2015] Durner, R. and Kellerer, W. (2015). The cost of security in the sdn control plane. *CoNEXT Student Workhop*.
- [Fonseca et al., 2012] Fonseca, P., Benesby, R., Mota, E., and Passito, A. (2012). A replication component for resilient openflow-based networking. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 933–939. IEEE.
- [Giotis et al., 2014] Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., and Maglaris, V. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Computer Networks*, 62:122–136.
- [Gogoi et al., 2011] Gogoi, P., Bhattacharyya, D., Borah, B., and Kalita, J. K. (2011). A survey of outlier detection methods in network anomaly identification. *The Computer Journal*, page bxr026.
- [Hommes et al., 2014] Hommes, S., State, R., and Engel, T. (2014). Implications and detection of dos attacks in openflow-based networks. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, pages 537–543. IEEE.
- [Hu et al., 2013] Hu, Y. L., Su, W. B., Wu, L. Y., Huang, Y., and Kuo, S. Y. (2013). Design of event-based Intrusion Detection System on OpenFlow Network. *Proceedings of the International Conference on Dependable Systems and Networks*, pages 4–5.
- [Kandoi and Antikainen, 2015] Kandoi, R. and Antikainen, M. (2015). Denial-of-service attacks in OpenFlow SDN networks. *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015*, pages 1322–1326.
- [Kloti et al., 2013] Kloti, R., Kotronis, V., and Smith, P. (2013). Openflow: A security analysis. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–6. IEEE.
- [Kreutz et al., 2013] Kreutz, D., Ramos, F. M., and Verissimo, P. (2013). Towards secure and dependable software-defined networks. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13*, page 55.
- [Kreutz et al., 2015] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking : A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14 – 76.

Bibliography

- [Kührer et al., 2014a] Kührer, M., Hupperich, T., Rossow, C., and Holz, T. (2014a). Exit from hell? reducing the impact of amplification ddos attacks. In *USENIX Security*, volume 2014.
- [Kührer et al., 2014b] Kührer, M., Hupperich, T., Rossow, C., and Holz, T. (2014b). Hell of a handshake: Abusing tcp for reflective amplification ddos attacks. In *WOOT*.
- [Laboratory, 1999] Laboratory, M. L. (1999). MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation.
- [Li et al., 2014] Li, J., Berg, S., Zhang, M., Reiher, P., and Wei, T. (2014). Draw Bridge-Software-Defined DDoS-Resistant Traffic Engineering. *ACM SIGCOMM Computer Communication Review (SIGCOMM 2014)*, 44(4):591–592.
- [McKeown et al., 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69.
- [Mirkovic and Reiher, 2004] Mirkovic, J. and Reiher, P. (2004). A taxonomy of ddos attack and ddos defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53.
- [Mohammadi et al., 2017] Mohammadi, R., Javidan, R., and Conti, M. (2017). Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks. *IEEE Transactions on Network and Service Management*.
- [Mousavi and St-Hilaire, 2015] Mousavi, S. M. and St-Hilaire, M. (2015). Early detection of ddos attacks against sdn controllers. In *Computing, Networking and Communications (ICNC), 2015 International Conference on*, pages 77–81. IEEE.
- [Patcha and Park, 2007] Patcha, A. and Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12):3448–3470.
- [Peng et al., 2007] Peng, T., Leckie, C., and Ramamohanarao, K. (2007). Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Computing Surveys (CSUR)*, 39(1):3.
- [Ranjan et al., 2006] Ranjan, S., Swaminathan, R., Uysal, M., and Knightly, E. W. (2006). Ddos-resilient scheduling to counter application layer attacks under imperfect detection. In *INFOCOM*.
- [Ranjan et al., 2009] Ranjan, S., Swaminathan, R., Uysal, M., Nucci, A., and Knightly, E. (2009). Ddos-shield: Ddos-resilient scheduling to counter application layer attacks. *IEEE/ACM Transactions on Networking (TON)*, 17(1):26–39.

Bibliography

- [Reitblatt et al., 2012] Reitblatt, M., Foster, N., Rexford, J., Schlesinger, C., and Walker, D. (2012). Abstractions for network update. *ACM SIGCOMM Computer Communication Review*, 42(4):323.
- [Scott-Hayward et al., 2016] Scott-Hayward, S., Natarajan, S., and Sezer, S. (2016). Survey of Security in Software Defined Networks. *Surveys & Tutorials*, 18(1):623–654.
- [Shanmugam et al., 2014] Shanmugam, P. K., Subramanyam, N. D., Breen, J., Roach, C., and Van Der Merwe, J. (2014). DEIDtect: Towards distributed elastic intrusion detection. *DCC 2014 - Proceedings of the ACM SIGCOMM 2014 Workshop on Distributed Cloud Computing*, pages 17–23.
- [Shin and Gu, 2013] Shin, S. and Gu, G. (2013). Attacking software-defined networks: A first feasibility study. *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 165–166.
- [Shin et al., 2013] Shin, S., Yegneswaran, V., Porras, P., and Gu, G. (2013). Avant-guard: scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 413–424. ACM.
- [Shirali-Shahreza and Ganjali, 2013] Shirali-Shahreza, S. and Ganjali, Y. (2013). Flexam: flexible sampling extension for monitoring and security applications in openflow. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 167–168. ACM.
- [Wald, 1973] Wald, A. (1973). *Sequential analysis*. Courier Corporation.
- [Wang et al., 2015] Wang, H., Xu, L., and Gu, G. (2015). Floodguard: A dos attack prevention extension in software-defined networks. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, pages 239–250. IEEE.
- [Wueest, 2014] Wueest, C. (2014). The continued rise of ddos attacks. *Security Response by Symantec version 1.0–Oct*, 21.
- [Xia et al., 2015] Xia, W., Wen, Y., Foh, C. H., Niyato, D., and Xie, H. (2015). A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, 17(1):27–51.
- [Xing et al., 2013] Xing, T., Huang, D., Xu, L., Chung, C. J., and Khatkar, P. (2013). Snort-Flow: A OpenFlow-based intrusion prevention system in cloud environment. *Proceedings - 2013 2nd GENI Research and Educational Experiment Workshop, GREE 2013*, pages 89–92.

Bibliography

- [Yan et al., 2015] Yan, Q., Yu, F. R., Member, S., Gong, Q., and Li, J. (2015). Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments : A Survey , Some Research Issues , and Challenges. 18(c):2–23.
- [Zargar and Joshi, 2013] Zargar, S. T. and Joshi, J. (2013). Dicodefense: distributed collaborative defense against ddos flooding attacks. In *IEEE symposium on security and privacy*.
- [Zargar et al., 2013] Zargar, S. T., Joshi, J., and Tipper, D. (2013). A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE communications surveys & tutorials*, 15(4):2046–2069.
- [Zhang et al., 2016] Zhang, P., Wang, H., Hu, C., and Lin, C. (2016). On denial of service attacks in software defined networks. *IEEE Networks*, 30(6):28–33.