ESCOLA POLITÉCNICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

CHARLES VARLEI NEU

DETECTING ENCRYPTED ATTACKS IN SOFTWARE-DEFINED NETWORKING

Porto Alegre

2019

PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER SCIENCE GRADUATE PROGRAM

DETECTING ENCRYPTED ATTACKS IN SOFTWARE-DEFINED NETWORKING

CHARLES VARLEI NEU

Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of Ph. D. in Computer Science.

Advisor: Prof. Dr. Avelino Francisco Zorzo

Porto Alegre
2019

Ficha Catalográfica

N477d Neu, Charles Varlei

Detecting Encrypted Attacks in Software-Defined Networking / Charles Varlei Neu . – 2019.
132.
Tese (Doutorado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Avelino Francisco Zorzo.

1. SDN. 2. Encrypted threats. 3. Security. 4. IDS/IPS. 5. Encrypted traffic classification. I. Zorzo, Avelino Francisco. II. Título.

Charles Varlei Neu

DETECTING ENCRYPTED ATTACKS IN SOFTWARE-DEFINED NETWORKING

This Thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on Jan 22, 2019.

I would like to express my sincere gratitude to several people who have contributed to the completion of this thesis. First, I thank my supervisor, Professor Avelino Zorzo, who has significantly contributed to my research. His expertise in computer security and many other subjects was fundamental to my learning, as well as the development of my research work and thesis.

I am also very much indebted to members of our research group (CONSEG/PUCRS) for the discussions we had while my PhD. Those discussions led me to work with the subject I present in this thesis.

Thanks to Professor Ina Shiering for receiving me and supervising my research during my stay at Ostfalia University (Germany). Thanks also to Professor Feng Hao for receiving me and supervising my research at Newcastle University (England) and also for helping me in being very practical with regard to my thesis. His observations while discussing the subject and results of my thesis were very important for its completion.

My sincere thanks for the support and love shown by my family and friends.

Finally, many thanks to the financial support provided by Brazilian Research Agency CAPES, German DAAD, Ostfalia University of Applied Sciences, Newcastle University SRS Research Group and TOTVS.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior – Brasil (CAPES) – Finance Code 001.

DETECTING ENCRYPTED ATTACKS IN SOFTWARE-DEFINED NETWORKING

ABSTRACT

Security is one of the major concerns for the computer network community due to resource abuse and malicious flows intrusion. Nowadays, cryptography is being widely used as a standard for securing data exchange on the Internet. However, attackers are improving methods by using encryption over malicious packets or flows so that it may be more difficult to being detected. Furthermore, those attacks are more effective on their malicious purposes when cryptography is used. Usually, before a network or a system is attacked, to perform a denial of service, for example, a port scan is performed to discover vulnerabilities, such as open ports. Several studies have addressed Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) methods for detecting and preventing attacks, based on flows or packet data analysis. However, typically those methods lead to an increase in switching latency, due to the need to analyze flows or packets before routing them. This may also increase network overhead when flows or packets are duplicated to be parsed by an external IDS. On the one hand, an IDS/IPS may be a bottleneck on the network and may not be useful, specially if traffic is encrypted. On the other hand, the new paradigm called Software-Defined Networking (SDN) provides statistical information about the network that may be used for detecting malicious activities. Hence, this work presents an approach for detecting encrypted malicious activity in SDN, such as port scan, denial of service and generic attacks, based on switch counters data. Thus, the developed methods are non-intrusive and lightweight, with low network overhead and low memory and processing power consumption. The results show that our methods are effective on detecting such attacks by discovering anomalies on the network activities, even when flows or packets are encrypted.

Keywords: SDN, encrypted threats, security, IDS/IPS, encrypted traffic classification, DoS/DDoS, port scan, encrypted attacks.

DETECTING ENCRYPTED ATTACKS IN SOFTWARE-DEFINED NETWORKING

RESUMO

A segurança é uma das principais preocupações da comunidade de redes de computadores devido ao aumento de fluxos maliciosos. Atualmente, a criptografia está sendo amplamente usada como padrão para a troca de dados segura na Internet. No entanto, os atacantes também estão utilizando a criptografia em seus ataques para dificultar a detecção e tornar os ataques mais eficazes em seus propósitos maliciosos. Normalmente, antes de atacar uma rede ou um sistema de computação, para executar um ataque de negação de serviços, por exemplo, uma varredura de portas é executada para descobrir vulnerabilidades. Vários estudos abordaram os sistemas de detecção de intrusão (Intrusion Detection Systems - IDS) e os sistemas de prevenção de intrusão (IPS - Intrusion Prevention System) para detecção e prevenção de ataques, com base na análise de dados de fluxos ou pacotes. Entretanto, normalmente, esses métodos levam a um aumento na latência para encaminhar os dados, devido à necessidade de analisar fluxos ou pacotes antes de roteá-los. Isso também pode aumentar a sobrecarga de rede quando fluxos ou pacotes são duplicados para serem analisados por um IDS externo. Por um lado, um IDS/IPS pode ser um gargalo na rede e pode não ser útil, especialmente se o tráfego for criptografado. Por outro lado, o novo paradigma chamado SDN (Software-Defined Networking) provê informações estatísticas sobre a rede que podem ser usadas para detectar atividades maliciosas. Assim, este trabalho apresenta uma abordagem para detectar atividade maliciosa criptografada em SDN, como varreduras de porta, negação de serviços e ataques genéricos, com base em dados de contadores dos swithes. Os resultados mostram que nossos métodos são eficazes na detecção de tais ataques, descobrindo anomalias nas atividades da rede, mesmo quando os fluxos ou pacotes são criptografados. Além disso, geram baixa sobrecarga de rede e necessitam pouco consumo de memória e processamento.

Palavras-Chave: SDN, ataques criptografados, segurança, IDS/IPS, classificação de trafego criptografado, DoS/DDoS, port scan .

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	- Application Programming Interfaces
ASIC	- Application-Specific Integrated Circuit
BGP	- Border Gateway Protocol
CA	- Certificate Authority
CE	- Control Elements
CSV	- Comma-Separated Values
CUSUM	- Cumulative Sum
DoS	- Denial of Service
DDoS	- Distributed Denial of Service
DFZ	- Internet default-Free Zone
DIDS	- Distributed Intrusion Detection System
DHCP	- Dynamic Host Configuration Protocol
DNS	- Domain Name Service
DPI	- Deep Packet Inspection
EC	- Exclusion Criteria
EID	- Endpoint Identifies
FE	- Forward Elements
ForCES	- Forwarding and Control Element Separation
FOS	- Fast Orthogonal Search
HIDS	- Host Intrusion Detection System
IC	- Inclusion Criteria
IDS	- Intrusion Detection System
IETF	- Internet Engineering Task Force
IoT	- Internet of Things
IPS	- Intrusion Prevention Systems
IP	- Internet Protocol
IT	- Information Technology
kNN	- k-nearest neighbor
LFB	- Logical Function Blocks
LISP	- Locator/ID Separation Protocol
ML	- Machine Learning

NBA	- Network Behavior Analysis
NBAR	- Network Based Application Recognition
NE	- Network Element
NETCONF	- Network Configuration Protocol
NIDS	- Network Intrusion Detection System
NSM	- Network Security Monitoring
OISF	- Open Information Security Foundation
OpFlex	- Open Policy Framework
POF-FIS	- POF Flow Instruction Set
PCA	- Principal Components Analysis
PL	- Protocol Layer
POF	- Protocol-Oblivious Forwarding
POF-FIB	- POF Flow Instruction Blocks
POF-FIS	- POF Flow Instruction Set
R2L	- Remote to User
REST	- Representational State Transfer
RL	- Reinforcement Learning
RLOC	- Routing Locators
RPC	- Remote Procedure Calls
RQ	- Research questions
RLOC	- Routing Locators
SIEM	- Security Information and Event Management
SNMP	- Simple Network Management Protocol
SSL	- Secure Sockets Layer
SNMP	- Simple Network Management Protocol
SDN	- Software-defined Networking
SMS	- Systematic Mapping Study
STRIDE	- Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege
TCP	- Transport Control Protocol
TLS	- Transport Layer Security
TML	- Transport Mapping Layer
U2R	- User to Root

CONTENTS

1. INTRODUCTION

Nowadays the increasing number of devices connected to the Internet, along with the high number of transactions performed online, augment the challenge to keep devices and transactions secure. Attackers are constantly evolving their methods, materializing different kinds of attempts to steal or damage data, damage computer environments or even disrupt the normal system or network behavior. Zero-day Denial of Service (DoS) attacks [NPSR16, MZR14], as well as Distributed Denial of Service (DDoS) [MZR14], become even more common in high-speed networks due to constantly increasing number of new vulnerabilities. Such attacks are even more dangerous if encryption is used by the attackers, since threats may bypass protection systems or consume more hardware resources, because it may take longer to analyze encrypted packets or flows [SKS15] [Gar].

Usually, when attackers plan to invade a system or a network, so that they may be able to launch a specific attack, they need some information about vulnerabilities that allow the intrusion to happen. Therefore, typically, a scan attack [NTL+18][HPSR18] is used. According to CERT [CER18], a Brazilian center that monitors incident reports, scan attempts represented more than 50% of all the reported incidents in 2017. Some protection mechanisms are typically used to improve security by providing prevention, detection and response. Prevention is the process of trying to stop intruders from access system or network resources. The detection process is related to discover if an attack has been performed, i.e., when the intruder has succeeded or is trying to access the system or network. The response process is used to fix the failure of the prevention and detection mechanisms, by stopping and even preventing new similar attempts [BZ17].

In the literature, several studies address Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) methods for detecting malicious activities, like port scan and DoS/DDoS, based on received flows or packet data analysis [YY15] [JGSG17] [MZR14] [NZOM16]. However, those methods lead to an increase in switching latency due to the need to analyze flows or packets before routing them, and the need for high processing power and memory consumption on high speed networks. This may also increase network overhead when flows or packets are duplicated to be parsed by an external IDS. In this context, an IDS/IPS may be a bottleneck on the network and may not be useful on a system or network. Moreover, those methods typically do not prevent new attack attempts, and, as they also frequently present some false negative detection and do not provide protection after a port scan is detected, an attacker may be able to perform an attack, like DoS/DDoS using encrypted packets so that the attack may be more powerful and also harder to be detected by traditional IDS/IPS. Thus, it is necessary to develop new methods, or rather improve available ones, specially to detect such threats in new network technologies, like Software-Defined

Networking (SDN)[OPE18b] [MVTG14] [GBC16], that brings new security challenges, but also new opportunities and resources to develop new protection tools[NZOM16].

SDN has emerged as a new paradigm on computer networks, and security may be improved since it offers new opportunities for developing IDS methods. SDN is managed by a centralized entity, called controller, and can use protocols, like OpenFlow [OPE18b, MVTG14], to provide features and resources that may be used by an IDS method. One such feature is the switch counters data stored in a counters table [NZOM16]. Moreover, as the controller has a global view on the network and may exchange information with other controllers that manage other networks, a more sophisticated protection system can be implemented, extending protection to the entire network as well as to other networks [NZOM16].

Therefore, this work presents an approach for detecting encrypted attacks on SDN and also to protect the network against such attacks. This approach is based on the switch counters data and intended specially to protect against port scan and DoS/DDoS, although protection against other attacks can be provided by adapting our proposed methods. These data are available to the controller and may be collected through specific messages on pre-defined time intervals to perform detection and then to block threats, being non-intrusive and lightweight. Furthermore, it does not overload the network and uses low processing power and memory. Thus, whenever an attacker performs a scan on the network, our IPS will detect it and then prevent new malicious attempts, like encrypted DoS/DDoS[1] application threats, improving security of SDN infrastructures. Moreover, we also have developed methods to identify encrypted DoS and show an approach for detecting generic attacks, even if the flows are encrypted.

1.1 Problem definition and objectives

The main goal of this research is to propose an approach to identify malicious encrypted traffic in Software-Defined Networking. To achieve this research objective, we derived the following objectives:

- Deepening the background concerning SDN, SDN protocols and SDN controllers;
- Studying the main encrypted attacks in SDN;
- Studying current approaches for encrypted traffic management and malicious encrypted traffic detection;
- To analyze available datasets for intrusion detection systems evaluation;
- Validating and evaluating the methods proposed through case studies;

[1]We use DoS for both DoS and DDoS throughout the remaining of this thesis.

- Documenting and reporting the study results, publishing them in scientific conferences and journals and make the source-code and datasets available to the community.

1.2 Summary of contribution and hypothesis

The results of our Systematic Mapping Study (see Chapter 3) show that currently Network IDS have some limitations and need to be improved. One such limitation is their inability to analyze encrypted traffic. Moreover, only few works that address protection methods for SDN were found, and they do not mention encrypted attacks. Thus, in this thesis we aim to address those limitations. Table 1.1 shows the thesis subject, research topic, the hypotheses to be validated, the research questions that will be answered and the main goals of this research.

Table 1.1 – Syntheses of the thesis

INPUT	**OUTPUT**
SUBJECT	Network Security
TOPIC	Encrypted attacks detection in SDN
HYPOTHESIS	It is possible to provide a lightweight solution to detect attacks in SDN with low hardware resource usage and without reducing network performance even when the flows are encrypted.
RESEARCH QUESTION	RQ1: What are the main attacks in SDN? RQ2: Which limitations current IDS have on detecting attacks in SDN? RQ3: How to protect SDN against malicious encrypted traffic?
MAIN GOAL	Develop a way to protect SDN against malicious encrypted traffic without overloading the network and with low resource usage.

To answer the research questions, to validate our hypothesis and to achieve our main goal in this thesis, experimental scenarios are used. These scenarios are simulated networks in specific tools that allow the deployment of SDN topology and the evaluation of the protection methods that are being implemented. Public available IDS datasets are also collected and analyzed to help in defining protection methods, as well as to be used for analyzing normal and abnormal behaviors and encrypted packets.

1.3 Thesis outline

This thesis is structured as follows. Chapter 2 brings an overview of the background that supports the main investigated topics. Initially, SDN, its protocols and controllers are described. After, an overview of cryptography and some cryptography algorithms are presented. Next, we describe some of the main attacks against SDN and traditional networks found in the literature. Later, we present definitions for IDS and IPS, their main operation forms, methods and datasets that are available. Finally, we present an overview of encrypted traffic management and the main approaches used to deal with encrypted traffic. Chapter 3 describes our Systematic Mapping Study related to network intrusion detection methods. The used methodology and outcomes are discussed, and how our research problem was defined. Chapter 4 presents an overview of the state of the art regarding encrypted attacks detection and encrypted traffic management. We describe methods related to SDN, as well as some methods intended to traditional network that may be adapted to SDN. Finally, we discuss the described approaches and their limitations.

The contributions of this thesis are described in Chapters 5 and 6. Chapter 5 presents our approaches to detect encrypted attacks in SDN. First, we describe what data are used to design our methods and how it is available and can be collected. Then, approaches for detecting generic anomalies in encrypted traffic is presented. After, we present our methods to detect specific attacks in encrypted SDN flows, which are port scan and DoS. Chapter 6 reports the results of our methods[2] based on a case study performed in a controlled environment. First, the used network topology and technologies are described. Then, we show a case study on this emulated network under a port scan attack. Later, we inject an encrypted DoS attack on the same environment and analyze how our system acts as a protection mechanism. Finally, Chapter 7 concludes the thesis, revisiting the main thesis contributions, describes limitations, sketches ongoing research and indicate some future directions.

[2]We use method and approach as synonyms throughout this thesis.

2. BACKGROUND

This chapter introduces the main topics related to the approach proposed in this thesis, *i.e.*, SDN, protocols, cryptography, IDS/IPS. It also explains the main attacks in SDN and also traditional networks found in the literature. Finally, network traffic (encrypted and not encrypted) classification methods are presented.

2.1 Software-Defined Networking

Computer networks became an essential part of modern live, being a core component, for example, in business, industry, health systems, security, education and entertainment. Advents like Internet of Things (IoT) are connecting every kind of electronic devices and control systems, increasing network requirements even more. This increasing number of devices that are communicating through networks, like the Internet, require new network models that are more flexible in configuration, implementations, management and more hardware independent. During the last decades, the network infrastructure and protocols evolution was restrained by some factors, such as vendor-specific technologies, compatibility with legacy technologies, and so forth. That network ossification is a significant barrier to innovation, and disruptive approaches are demanded to offer alternatives to this stagnation. Thus, traditional networks technologies are not sufficient to supply current communication needs. Thus, the SDN concept arises as an opportunity to break the *status quo*, through the separation of control logic from the underlying network devices. In traditional networks the data plane, control plane and some applications are embedded in network devices. This arrangement is separated in SDN, where the control plane and applications are logically centralized [NTL+18] [KREV+15] [Shu13].

According to the Open Networking Foundation [Ope18a], the SDN architecture (Figure 2.1) can be split into three functional layers: Application, Control and Infrastructure:

- Application plane (Application layer): is responsible for providing end-user networking management, analytic and business applications that consume the SDN communication and network services, such as load balancing and firewall.
- Control plane (Control layer): is composed by a set of controllers that provide control functionality, receiving instructions or data from the Application plane and relaying them to the Data plane. It also extracts information from the Data plane (such as statistics and events) and communicates that back to the Application plane.
- Data plane (Infrastructure layer): is composed by networking devices that control the forwarding and switching for the network, including physical and virtual switches. Both

Application and Control planes communicate through a Northbound interface. Likewise, Control and Data planes communicate through a Southbound interface.

Figure 2.1 shows a comparison between traditional network architecture and SDN architecture.

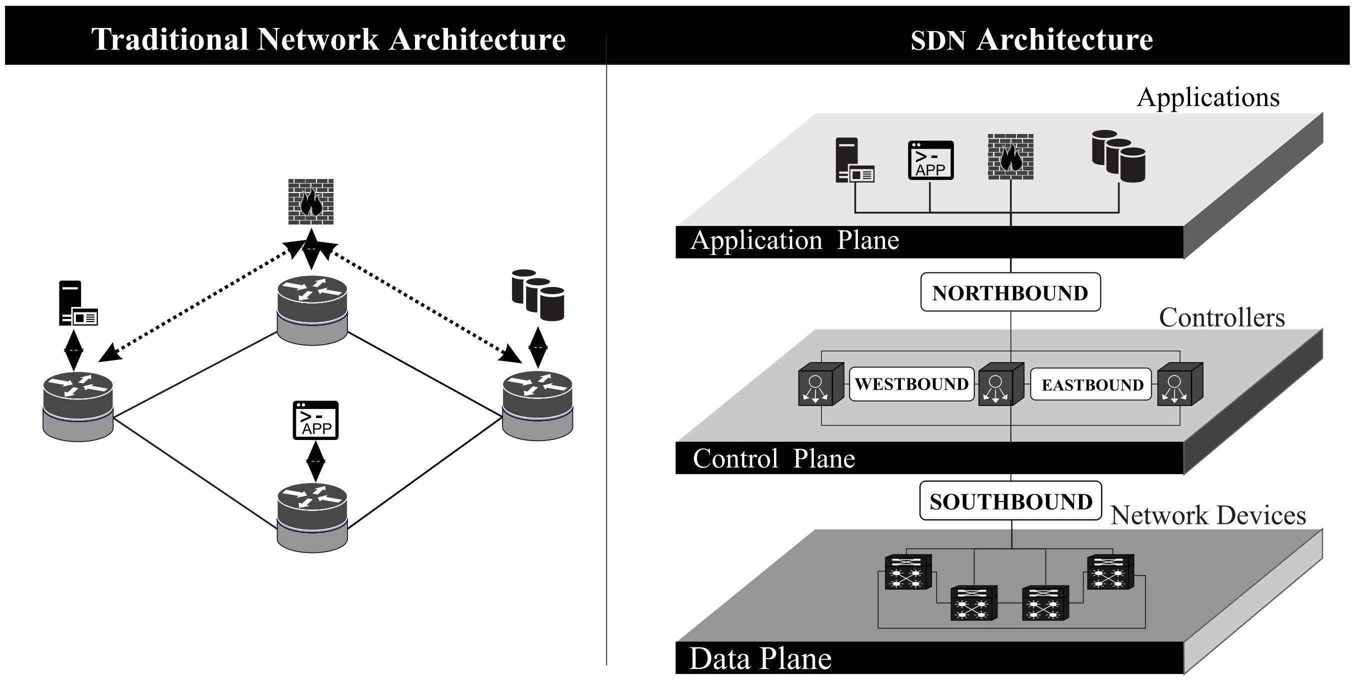

Figure 2.1 – Traditional network architecture vs SDN architecture

To provide and to optimize communication between the Control layer and the other layers and to allow the SDN controller to be able to communicate with other controllers, the SDN architecture is also separated into three main communication mechanisms: Northbound, Eastbound/Westbound and Southbound [GBC16] [Ope] [Fou11]. Each kind of interface enables different communication:

- Northbound interfaces: provide communication between the Application layer and the controller. It also allows the connection with automated stacks used for Cloud management, such as OpenStack [Ope18c] [KXF18] and CloudStack [Clo]. Currently, the Representational State Transfer (REST) protocol [MZR14] is the most used in the Northbound interface, being implemented by the main available SDN controllers [SEKC16].
- Eastbound/Westbound interfaces: provide communication for managing a distributed SDN architecture, with several controllers sharing management data.
- Southbound Interfaces: Also known as control/data plane interface [HHB14], provides Application Programming Interfaces (API) that are used for network control. The SDN controller uses them to create and change forwarding rules in the Data plane devices, like switches and routers. The Southbound interface is performed by some protocols, such as NETCONF [GBC16], SNMP [Pre], OVSDB [IET13], I2RS [IET16], Border

Gateway Protocol (BGP) [CLLL16], LISP [RNME+15], XMPP [NS14], Control Element Separation (ForCES) [HSDK15], OpFlex [SDYL+], and OpenFlow [MAB+08].

2.1.1 SDN protocols

Several protocols are available for communication in SDN. This section describes some of the most used protocols for Southbound communication in SDN [GBC16]. OpenFlow is currently a standard in SDN controllers and switches communication and is widely used in the related SDN works (see Chapter 4). Therefore, OpenFlow will also be used in the most experiments in this Thesis. However, other protocols are emerging as promising alternatives and will also be described in this Section.

OpenFlow protocol

The OpenFlow protocol is intended to enable communication between an SDN Controller and switches. It is an extensible protocol that provides mechanisms that can be used to define additional protocol elements by network developers and managers, such as new match fields, specific actions and port properties, to improve new network technologies, collect network information, identify network issues and to define network behaviors. OpenFlow also allows switch and controller vendors to work independently to create interoperable SDN devices. The first version of OpenFlow (version 1.1) was released in February 2011 and the project is currently maintained and developed by the Open Networking Foundation [Fou11]. OpenFlow enables a controller to make routing decisions periodically or in an *ad hoc* manner, as well as translating them into rules and actions, by using a switch flow table. Packets that are unmatched by the switch can be forwarded to the controller, which can then decide to modify existing switch flow table rules or to define new forwarding rules [Fou11]. Currently, the OpenFlow protocol is a *de-facto* open SDN Southbound messaging standard protocol that supports three message type modes, in which each one contains multiple subtypes [Fou14] [Ope][Foub] [Fouc]:

- Controller-to-switch messages: used to manage or inspect the state of the switch and they are initiated by the controller. It may or may not require a response from the switch. The main messages are for features request/response, configuration request/set, modify-state, multi-part (used for collecting statistics from switches flow table) and barrier request/reply.
- Asynchronous messages: used to update the controller of network events and changes to the switch state, they are initiated by the switch to inform packet arrival, switch state change, or errors. The four main messages are packet-in, flow-removed, port-status and error.

- Symmetric messages: are initialized by the switch as well as by the controller and are sent without a previous request. The main messages are Hello and Echo request/reply.

OpenFlow works based on three basic components: an OpenFlow-enabled switch, a communication channel and a controller. This switch uses a flow table pipeline to forward a sequence of packets, as shown in Figure 2.2. When a packet arrives on the switch (Packet in), it tries to match the packet with rules in the flow table to perform some action, like forward the packet or even drop it.

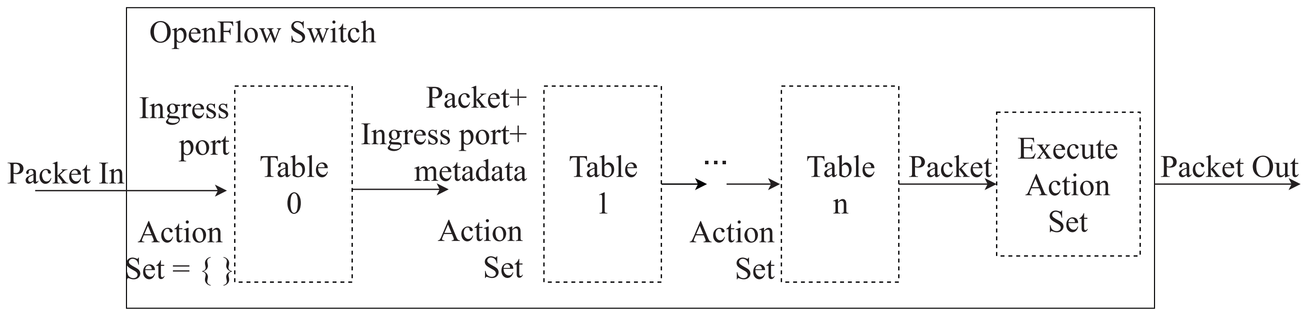

Figure 2.2 – OpenFlow switch pipeline

An OpenFlow switch is composed of flow tables that are responsible for maintaining information about flows. Each flow entry contains match fields, counters and a set of instructions that can be used to match arriving packets information [Fou14] [Foua]. A flow table consists of flow entries, as shown in Figure 2.3. Each flow entry's field data are used to perform specific tasks, such as [Foua]:

- Match fields: which are used to match packets, based on the ingress port and packet headers. Metadata specified by a previous table can also be used.
- Priority: used for matching precedence of the flow entry.
- Counters: which are updated when packets are matched and provide statistical data that can be used for network management and activity analysis.
- Instructions: used to set or change actions or pipeline processing.
- Timeouts: which are time thresholds to consider flows as expired by the switch.
- Cookie: which consists of opaque data value chosen by the controller and it is used by the controller to filter flow statistics, flow modification and flow deletion.

Match fields	Priority	Counters	Instructions	Time out	Coockie

Figure 2.3 – Flow table entries

A flow can be dropped and actions can be taken during the whole pipeline. In order to keep the flow table updated, flow entries can be removed via a request message from the controller, by the switch flow expiry mechanism or by using the switch eviction mechanism [Foua] [Fou11]. Figure 2.4 shows a flow rule structure.

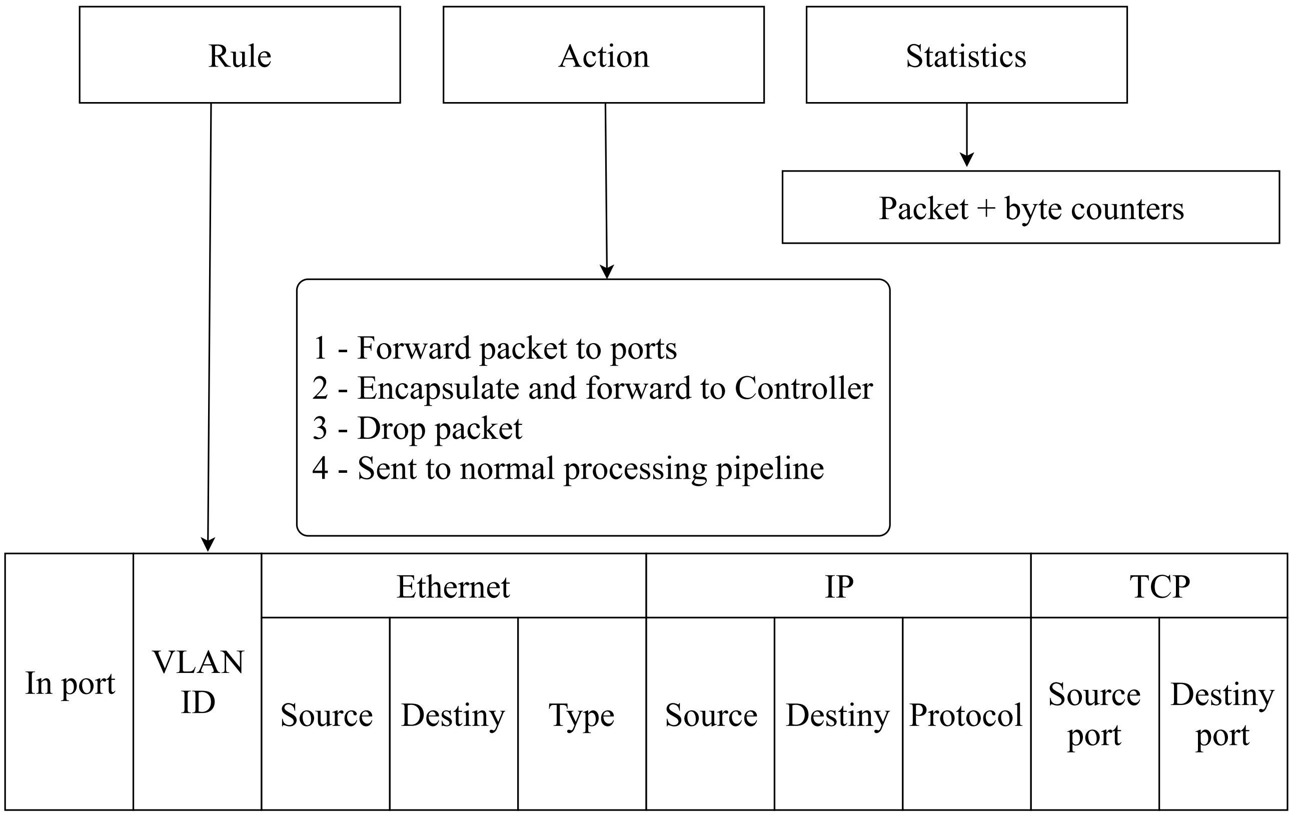

Figure 2.4 – Flow rule structure

When a packet comes in, the OpenFlow switch performs some functions, such as table lookup in the first flow table. This process can be extended to perform table lookups in other flow tables according to network requirements and configuration. Therefore, packet match fields are extracted from the packets, based on different header fields, such as Ethernet source address or IP destination address. In addition, matches can also be performed against the ingress port and metadata fields. Figure 2.5 [1] shows a packet flow matching process through an OpenFlow switch.

The OpenFlow switch specification supports a set of counters, like transmitted and received packets, flow duration, received and transmitted drops and errors, collisions and flow count [Ope18a, NZOM16]. Counters are maintained per table, per flow, per port and per queue. Those counters can be collected periodically by a specific job in the controller by using OFPT_STATS_REQUEST and OFPT_STATS_REPLY messages (OpenFlow v1.0, v1.1, v1.2 and v1.3) or OFPT_MULTIPART_REQUEST and OFPT_MULTIPART_REPLY (OpenFlow v.14 and v1.5) messages. Table 2.1 presents some of the counters available in the flow table, being "O" Optional and "R" Required counters [Ope18a].

[1]We use the standard symbols provided by draw.io to draw our flowcharts throughout this thesis

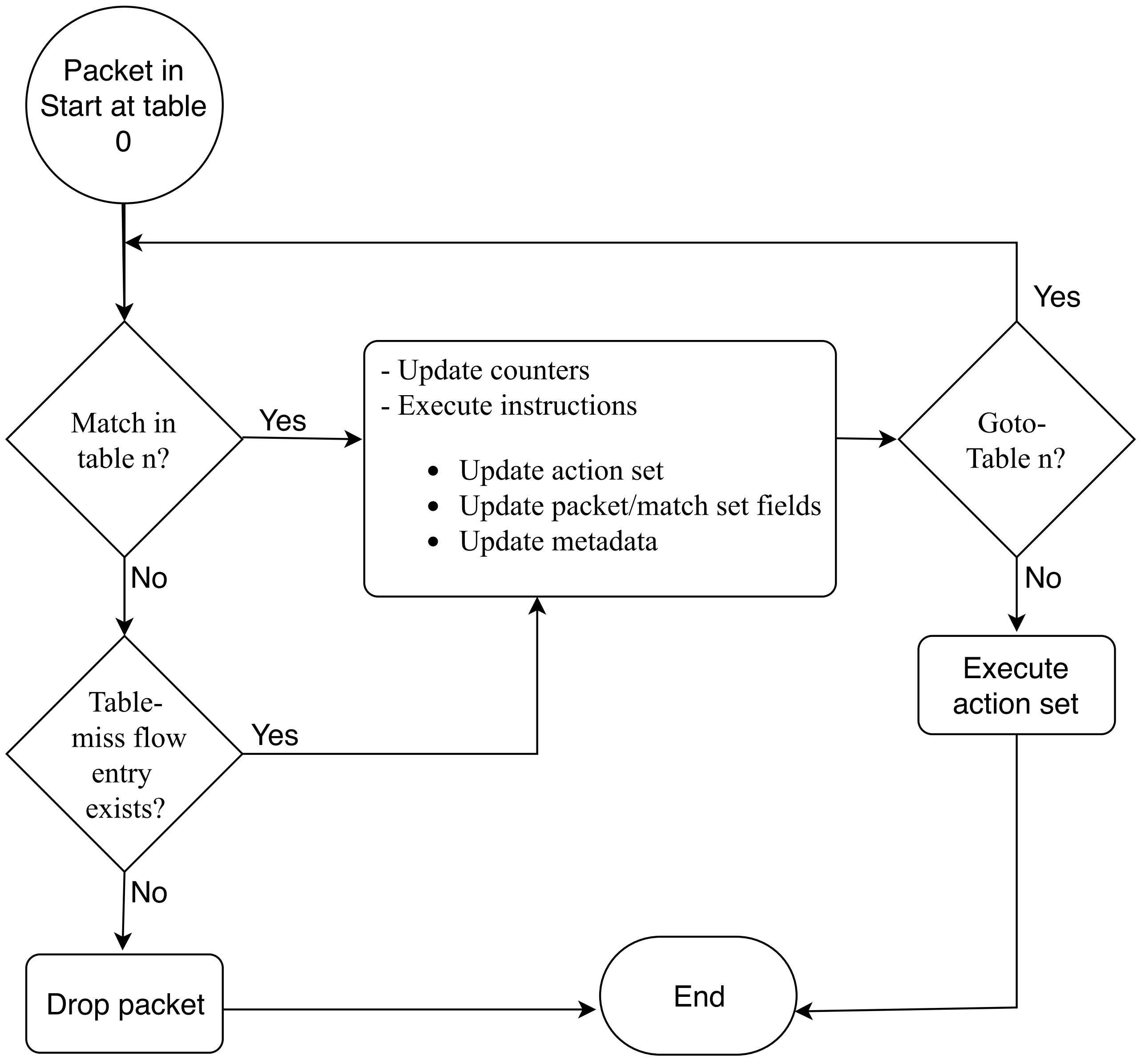

Figure 2.5 – Packet flow through an OpenFlow switch

Network Configuration Protocol

The Network Configuration Protocol (NETCONF) [PTL15] is a network management protocol that was developed by the NETCONF working group and published in 2006 as RFC 4741 [Grob] and later revised in June 2011 and published as RFC 6241 [IET11]. It was developed to fulfill lacks on previous protocol (for example, Simple Network Management Protocol (SNMP)) on providing operators, as they were originally designed for device configuration and network monitoring only. Since the use of heterogeneous devices, which can also be provided by different vendors, in the same network became very common in networks, its management can be even more difficult for the operators. Yet Another Next

Table 2.1 – OpenFlow counters (values measured in bits)

SOURCE	COUNTERS
Per Flow Table	Reference count (R - 32) Packet lookups (O - 64) Packet matches (O - 64)
Per Flow Entry	Received packets (O - 64) Received bytes (O - 64) Duration in seconds (R - 32) Duration in nanoseconds (O - 32)
Per Port	Received/Transmitted packets (R - 64) Received/Transmitted bytes (O - 64) Receive/Transmitted drops(O - 64) Receive/Transmitted errors (O - 64) Receive frame Alignment errors (O - 64) Receive overrun errors (O - 64) Receive CRC errors (O - 64) Collisions (O - 64) Duration in seconds (R - 32) Duration in nanoseconds (O - 32)
Per Queue	Transmitted packets (R - 64); Transmitted bytes (O - 64) Transmitted overrun errors (O - 64) Duration in seconds (R - 32) Duration in nanoseconds (O - 32)
Per Group	Reference count of flow entries (O - 32) Packet count (O - 64) Byte count (O - 64) Duration in seconds (R - 32) Duration in nanoseconds (O - 32)
Per Group Bucket	Packet count (O - 64) Byte count (O - 64)
Per Meter	Flow count (O - 32) Input packet count (O - 64) Input byte counter (O - 64) Duration in seconds (R - 32) Duration in nanoseconds (O - 32)
Per Meter Band	In band packet count (O - 64) In band byte count (O - 64)

Generation (YANG) is a modeling language designed for NETCONF that provides an object-oriented approach and a human-readable format that helps on describing the data model that is used in the NETCONF protocol [SAS+17].

Operators can use NETCONF, for example, to get and to set device configuration data. Those data are encoded by using eXtensible Markup Language (XML) and transmitted via Remote Procedure Calls (RPC). NETCONF configuration data are stored in three different data stores [PTL15]:

- Running: contains the settings currently in use by the network device.

- Candidate: this data store is an optional device functionality that stores a set of configuration data that the controller can use to update the Running store data, thus changing the devices operation.

- Startup: contains the configuration data used on device starting operation.

The communication between controller and network devices is made using the client-server paradigm, with the controller acting as client and the network device acting as server. As NETCONF does not provide a standard mechanism for device configuration, each vendor can implement it in a different way. Therefore, it must be implemented on the controller to provide interoperability among different devices on the network. However, NETCONF provides a set of standard messages to exchange configuration data between the controller and devices, such as *get-config* and *edit-config*. These messages can be used, for example, to request device configuration data stored in the Running store [PTL15].

Locator/ID Separation Protocol

Locator/ID Separation Protocol (LISP) was originally designed to solve the scalability issues of the Internet default-free zone (DFZ) routing tables by pushing traffic engineering practices to the identifiers space. It splits current IP addresses overlapping semantics of identity and location into two separate namespaces: endpoint identifiers (EID) that are used to identify hosts, and routing locators (RLOC) that are used to route packets. Although the EID and RLOC can be defined as desired, for example MAC addresses, the official documentation suggests the usage of identifiers that are syntactically identical to current IPv4 and IPv6 addresses [RNME+15].

LISP can be used to decouple Data and Control planes, to provide network programmability and to centralize control through a mapping system and several components. Thus, any existing IP network can incorporate common SDN resources by upgrading some routers to the LISP tunnel routers and connecting them to a mapping system. Moreover, as LISP can be implemented incrementally, it allows the implementation of hybrid networks, *i.e.*, to combine SDN and traditional networks. Thus, any existing IP network can incorporate common SDN resources simply upgrading some routers to the LISP tunnel routers and connecting them to a mapping system. Besides, in contrast to the common SDN protocols designed to operate primarily in a single domain, LISP allows SDN policies to be applied in all domains if required. LISP elements also support the deployment of a programmable SDN in a transit network, such as the Internet, which is a more complex task to perform by using traditional SDN protocols [RNME+15].

Forwarding and Control Element Separation

Forwarding and Control Element Separation (ForCES) [HSDK15] is composed by a framework and a protocol that aim to standardize the communication between Control plane, which has Control Elements (CE), and Data plane, which has Forward Elements (FE). Therefore, ForCES defines an entity called Network Element (NE), which is composed by one or more CEs and one or more FEs that communicate by the ForCES protocol. The distribution of NEs can be within a single device (local) or distributed in several devices. Thus, ForCES provides a more flexible network management approach than traditional ones, without changing the current architecture of the network. Therefore, an external controller that is logically centralized is not required.

The Control and Data planes are separated, but can be kept in the same NE. FEs and CEs can be developed in hardware or implemented virtually, and each virtual FE is created using several Logical Function Blocks (LFB), which are interconnected components. Each LFB implements a specific function, being able to receive, transmit or modify packages. The LFB are structured by a XML description of all information that can be exchanged between the FEs and the CEs, so that an administrator can dynamically change and control packet processing rules and policies.

Two layers are provided for the communication between FEs and CEs: Transport Mapping Layer (TML) and Protocol Layer (PL). The TML uses protocols like SCTP, IP, TCP, UDP, ATM and Ethernet and provides the communication channel. The PL controls the messages being exchanged in the communication, and uses three communication channels between the FE and CE with different priorities (high, medium and low). These priorities are used to define the message set to be transmitted on each communication channel. The FoRCES protocol messages have a common header where basically message type, Source ID and Destination ID of the communicating elements are defined followed by the message body. ForCES provides several types of messages, which are:

- Association messages: Setup, Setup Response, Teardown.
- Configuration messages: Config, Config Response.
- Query messages: Query, Query response.
- Event Notification.
- Packet Redirect.
- Heartbeat.

Open policy framework

The Open Policy Framework (OpFlex) [IET] is another interface proposal, similar to ForCES. It is a policy-driven system that was designed to control a large set of physical and virtual devices and aims to improve scalability by distributing the complexity of network management back to the forwarding devices. This control protocol provides bidirectional communication of policy, events, statistics and faults, by using XML, JSON and RPC over TCP, for example.

The OpFlex architecture provides a distributed control system based on a declarative policy information model and offers a standardized, open source mechanism for transferring abstract policies between a controller and devices. This architecture is composed by:

- Policy repository: which is a logically centralized entity that defines the system policies.
- Policy element: which is a logical abstraction for a physical or virtual device that implements and applies the policies. These elements are used for requesting parts of the policy when new devices are connected, disconnected or changed state.
- Endpoint Registry: which stores the current operation status of each device that is connected to the network (endpoints). It receives data about each endpoint from the policy element and can store it locally or in a distributed way.
- Observer: this component is used to collect statistics, failures and events of each policy element through specific messages:
  - Identity: which is the first message between a policy repository and a policy element.
  - Policy Resolution: returns a set of policies from a policy repository to a policy element.
  - Policy Update: used by the policy repository to send information about changes in policies to a policy element.
  - Policy Trigger: used by a policy element for policies resolution.
  - Endpoint Declaration: used by the policy elements to update the endpoint registry about new endpoints attached to the network.
  - Endpoint Request: used to retrieve a set of identifies, such as the MAC address, from endpoints. This message is sent from policy elements to an endpoint registry.
  - Endpoint Policy Update: this message is used by the endpoint registry to inform changes in the endpoint policies.
  - State Report: this kind of messages is used by the policy elements to the observer with data about failures, other events and statistics.

Protocol-Oblivious Forwarding

Protocol-Oblivious Forwarding (POF) is an enhancement to OpenFlow-based SDN forwarding architecture [YWS+14] [LHA+18] that aims to improve the SDN programmability. POF enables the development of new protocols and forwarding services that the SDN Data plane can support without modifying network elements. A POF-enabled SDN network architecture is very similar to an OpenFlow-enabled architecture, composed of a Control plane that has a centralized controller and manages the switches in the forwarding plane based on flow tables. However, POF enables a more generic packet field description scheme for flow matching and processing.

Yu *et al.* [YWS+14] proposed a basic POF Flow Instruction Set (POF-FIS) that can be used to parse, edit and forward packets as designed and required by the controller. POF-FIS is in fact an enhancement of the instructions and actions defined in OpenFlow I.x [Foub]. Table 2.2 shows a basic POF Flow Instruction Set (POF-FIS), proposed by Yu *et al.* [YWS+14].

Table 2.2 – A basic POF flow instruction set

CATEGORY	**INSTRUCTIONS**
Editing	SET_FIELD, ADD_FIELD, DEL_FIELD, ALG, CALCULATE_CHECKSUM, SET_FIELD_UPDATE_CHECKSUM, INC_FIELD, DEC_FIELD, AND_FIELD, OR_FIELD. SRL_FIELD, SLL_FIELD, XOR_FIELD, NOR_FIELD, NOT_FIELD
Forwarding	GOTO_TABLE, COUNTER, OUTPUT, GROUP, MOVE_PACKET_OFFSET, SET_PACKET_OFFSET
Entry	SET_TABLE_ENTRY, ADD_TABLE_ENTRY, DEL_TABLE_ENTRY
Jump	BRANCH, COMPARE, JUMP
Flow	SET_FLOW_METADATA, GET_FLOW_METADATA, ORDER_ENFORCE

Each group of instructions is used to perform some specific task [YWS+14]:

- Editing: these instructions are used to edit packet data, which is very useful in the forwarding process, as protocol rules usually need to edit the packet data, for example, by writing, storing, copying and calculating. The most useful editing instructions are ADD_FIELD and DELETE_FIELD (to insert or delete fields into or from the packet) and SET_FIELD (sets packet data fields to any value, such as MAC address in Ethernet headers).

- Forwarding: this set of instructions is used for packet forwarding. The entire forwarding process for one packet in the network element can be composed by multiple processes that can be separated into different flow tables according to the functionality. Thus, the GOTO_TABLE instruction could be used to send the packet data from the previous flow table to the next flow table, for example. The COUNTER instruction can be used to count the number of packets that have already been handled, as well as count the total length in byte unit, which is useful for storing and collecting flow statistics.

- Entry: These instructions allow the network elements to operate the flow entry by itself. For example, the SET_TABLE_ENTRY instruction can be used to set the parameter and the match information of flow entries. ADD_ TABLE_ENTRY and DEL_TABLE_ENTRY are useful to delete or insert new flow entries into a flow table.
- JUMP and FLOW: JUMP instructions are used to change the packet data processing procedure and FLOW instructions can be used to perform some operations on the global status of the data flow.

POF-FIS was developed with the aim to provide flexible protocol rules implementation and to enable programmers and users to deploy services on a more easy way. POF-FIS is independent of target platform, Northbound interface, and the high-level programming language (programmers can use, for example, P4, C and Java to implement the forwarding process [YWS+14]. POF handles network flows by flow instructions based on POF Flow Instruction Blocks (POF-FIB), which are deployed by the controller through the POF Southbound interface. POF controllers can be handled by Command-Line Interfaces, Graphical User Interface and high-level programming language. SDN users can organize the POF-FIS to POF-FIBs using the Northbound interfaces, and then download the POF-FIBs to the network elements to design the forwarding application [YWS+14] [LHA+18].

2.1.2 Controllers

In SDN, intelligence, control and management is logically centralized in a single entity called controller. The controller is a core component in SDN that is responsible for making network decisions, such as add or remove entries in the switches flow table. It also enables a physical abstraction, which simplifies the application development and services that are related to network flow management. Thus, the SDN controller serves as a sort of network operating system and acts as a network management software that centralizes the management and control tasks, including, for example, security tools like IDS/IPS. This centralization offers several advantages, such as [KREV+15]:

- Optimizes changing network policies through high-level programming languages and software components. This simplifies the process and also improves accuracy, being less error prone.
- A control program can automatically react to spurious changes in network state, keeping high-level policies intact.

- The centralization of control logic in a controller with global knowledge of the state of the network, simplifies the development of more sophisticated functions, services and network applications.
- Allows to react automatically to any forged change on the network state, such as an attack, keeping the high-level policy straightforward.
- Optimizes network security, by centralizing protection mechanisms that allows a quick update of detection and protection rules that can be extended automatically to the whole network, as well as to other networks through a East/Westbound communication with other controllers.

In order to build a new SDN network, it is possible to use an existent controller or even customize one. There are several controllers found in the literature. Khondoker *et al.* [KZMB14] and Salman *et al.* [SEKC16] present a comparison of some of the main available controllers, such as: Pox [McC18], RYU [RYU18], Floodlight [Pro18] and OpenDaylight [OPE18b].

Table 2.3 – A comparison of some of the main SDN Controllers

	POX	**RYU**	**FLOODLIGHT**	**OPENDAYLIGHT**
Interfaces	SB (OpenFlow)	SB (OpenFlow) + SB Management (OVSDB JSON)	SB (OpenFlow) NB(Java REST)	SB (OpenFlow Others SB Protocols) NB (REST Java RPC)
Virtualization	Mininet Open vSwitch	Mininet Open vSwitch	Mininet Open vSwitch	Mininet Open vSwitch
GUI	Yes	Yes (Initial Phase)	Web UI (Using REST)	REST
REST API	No	Yes (For SB Interface Only)	Yes	Yes
Productivity	Medium	Medium	Medium	Medium
Open Source	Yes	Yes	Yes	Yes
Documentation	Poor	Medium	Good	Medium
Language Support	Phyton	Phyton-Specific + Message Passing Reference	Java + Any language that uses REST	Java
Modularity	Medium	Medium	High	High
Platform Support	Linux, Mac OS and Windows	Most Supported on Linux	Linux, Mac, Windows	Linux
TLS Support	Yes	Yes	Yes	Yes
OpenFlow Support	OF v1.0	OF v1.0 v2.0 v3,0 Nicira Extensions	OF v1.0 and above	OF v1.0 and above
OpenStack Networking (Quantum)	NO	Strong	Medium	Medium

2.1.3 Packet processing in SDN

By using a specific protocol in SDN, such as OpenFlow, and a specific switch, like OpenvSwitch, some features are provided and packet/flows process is defined. However, customization in those processes is usually not possible. For example, different flow counters (statistics) are defined, but the implementation of new ones, if required, is not enabled. However, new technologies are emerging to overcome this limitation. Programming Protocol-Independent Packet Processors, or P4 [P418], is a high-level language that allows to program the Data plane of network devices, *i.e.*, to program protocol-independent packet processors [BDG+14]. Thus, using P4, it is possible to define how packets are processed by the Data plane of a programmable forwarding element such as a hardware or software switch, network interface card, router or network appliance. P4 is designed to specify only the Data plane functionality of the target and works together with SDN control protocols, like OpenFlow. P4 programs may also be designed to partially define the interface by which the Control plane and the Data plane communicate. However, the Control plane functionality of the target cannot be described using P4.

SDN protocols, like OpenFlow, explicitly specifies protocol headers on which it operates. However, this set has grown in a short time frame, increasing complexity of the specification as well as still not being flexible to add new headers. P4 aims to solve this inflexibility by providing:

- Reconfigurability in the field: allowing to change the way switches process packets once they are deployed.
- Protocol independence: switches should not be tied to any specific network protocol.
- Target independence: programmers can describe packet processing functionality independently of the underlying hardware specification.

Furthermore, in a traditional switch Data plane, functionality is defined by the manufacturer. In contrast, the Control plane manages the Data plane by handling entries in tables, like routing tables. It also allows configuring specialized objects, like meters, and processing control packets or asynchronous events, such as link state changes or learning notifications, as shown in Figure 2.6.

In a nutshell, a P4 switch differs from a traditional switch in two basic concepts:

- Data plane functionality is defined by a P4 program. The Data plane is configured at initialization time to implement the functionality described by the P4 program and has no built-in knowledge of existing network protocols.
- Control plane communicates with the Data plane by using the same channels as in a fixed-function device. However, the set of tables and other objects in the Data plane

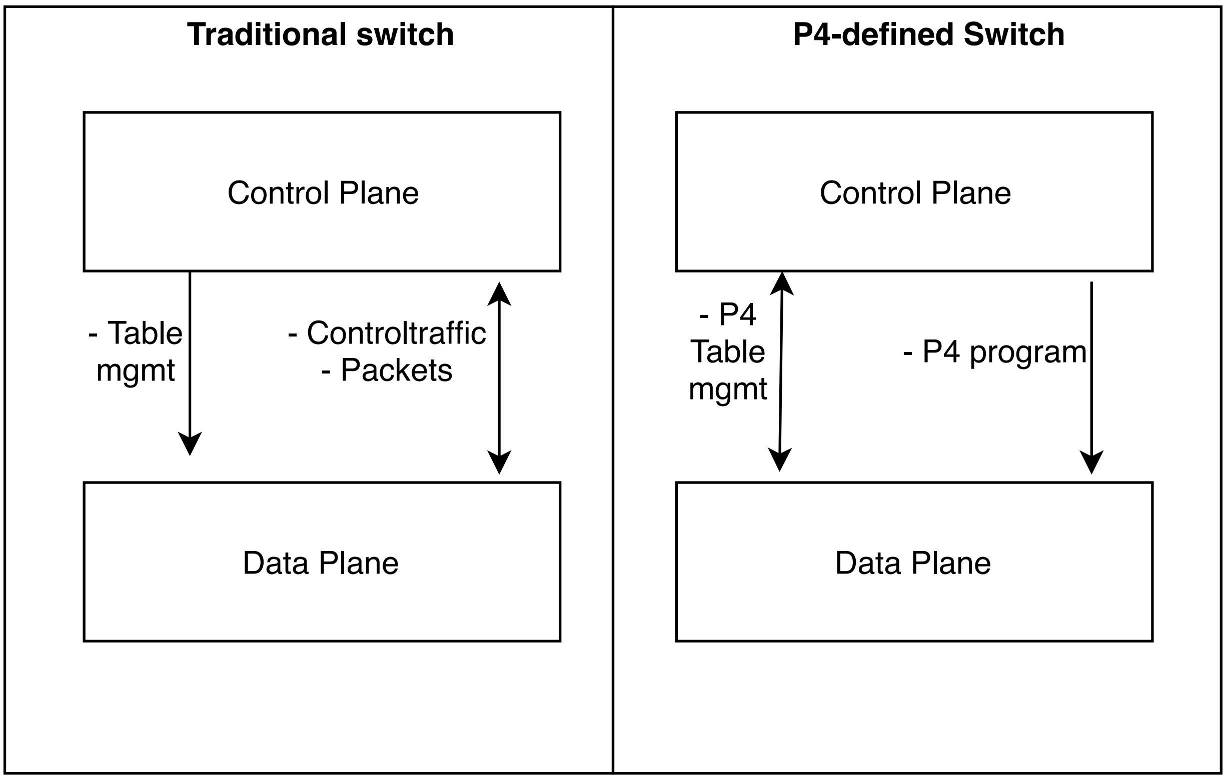

Figure 2.6 – Traditional switch vs P4-defined switch

are defined by a P4 program. The P4 compiler generates the API that the Control plane uses to communicate with the Data plane.

The main core abstractions that P4 can provide are:

- Header types: describe each header format within a packet, such as fields and sizes.
- Parsers: describe the sequences of headers within received packets, how to identify them and the headers and fields to extract from packets.
- Tables: describe the association of user-defined keys with actions. Allows to generalize traditional switch tables, enabling the programmer to implement routing tables, flow lookup tables, access-control lists, and other user-defined table types, even with complex multi-variable decisions.
- Actions: describe how packet header fields and metadata are manipulated.
- Match-action units: construct lookup keys from packet fields or computed metadata, perform table lookup using the constructed key, choose and execute actions.
- Control flow: describes packet-processing on a target, including the data-dependent sequence of match-action unit invocations. It also can perform deparsing (packet reassembly) using a control flow.

- External objects: these are architecture-specific constructs that can be manipulated by P4 programs through APIs.
- User-defined metadata: these are basically user-defined data structures associated with a packet.
- Intrinsic metadata: metadata provided by the architecture associated with each packet, such as port number where a packet has been received in.

P4 was designed to be protocol independent, and enables programmers to express a set of protocols and other Data plane behaviors according to the requirements. Thus, some inflexibility present on SDN Southbound protocols may be solved by using P4, such as, add new flow counters to compute switch flow statistics that can be requested by the controller. This can be useful, for example, for developing new IDS methods or even improve them.

2.2 Cryptography

This section describes some basic concepts on cryptography, which are the most used encryption protocols and why cryptography is used for securing data on communication through networks, specially on the Internet.

Cryptography is the science of writing secret code. The messages to be encrypted, called plaintext, are transformed by a function that is parameterized using a key. The result of this encryption process is the ciphertext, which will be the transmitted data. An intruder may be able to hear and to accurately copy the complete ciphertext, but as the intruder does not know the decryption key, data will still be protected [FS03] [KR16] [VMC08] [NZOM16].

According to Kumar *et al.* [KR16] and Ferguson *et al.* [FS03], a lot of encryption techniques are used on communications to provide security. Encryption algorithms can be categorized in two main groups:

- Symmetrical Encryption: only one secret key is used both to cipher and decipher messages. Thus, sender and receiver should know the secret key. Some examples of symmetric encryption are Blowfish, AES, RC4, DES, RC5, and RC6. AES implementations, such as AES-128, AES-192, and AES-256, are the most widely used symmetric algorithms nowadays.
- Asymmetrical Encryption: also known as public key cryptography, uses two different keys to encrypt a plaintext. The first is a public key, which is public available to anyone that wants to send a message. The second one is a private key, which is kept secret and only the owner knows it. This private key is used to decipher a message. Thus, a message that was encrypted by using a public key can only be deciphered using

a private key, and a message encrypted using a private key can be deciphered using a public key. Asymmetrical algorithms include ElGamal, RSA, DSA, Elliptic curve techniques and PKCS.

Asymmetric encryption is typically used in day-to-day communication channels. Encryption aims to ensure that four information security principles will be respected:

- Privacy: only the authorized recipient can read the message content, *i.e*, to understand a message the decipher key is required.
- Authentication: the recipient must be able to identify the sender and verify that it was him who sent the message. It proves their identities.
- Integrity: the recipient must be able to determine that the message has not been modified or altered from its original form during the transmission.
- Non-repudiation: ensure that the sender cannot deny the authorship of the message and the message was received by the specified person.

Homomorphic encryption [BsC+16] is a promising way of providing security to the data in which operations can be done on encrypted data itself, *i.e.*, encrypted data can be evaluated without decryption. Such schemes could help improving data confidentiality and integrity, and could be adopted in several real applications, such as in public clouds, financial, medical and genome data. However, researchers are still working to improve such schemes due to its noise growth and complexity [BsC+16] [RV17].

2.3 Attacks on SDN

SDN is a new paradigm on computer networks, nonetheless it also contains old problems and challenges that are present on other network technologies, specially related to security. An attack can be defined as a set of actions that attempt to consume computer system resources or to steal (or damage) any kind of data, regardless of whether successful or not. A common attack that can lead to system crashes is DoS [MZR14]. An intrusion attack can be defined as a set of actions that attempt to commit resources of a computer system or numerous attempts to exploit any kind of information, regardless of whether successful or not [BZ17]. A typical attacks classification could be as follows [BZ17] [PAGT07]. This classification is usually applied to traditional networks, but could also be extended to SDN.

- Denial of service (DoS): an attacker overloads some computing resources so that it can not handle legitimate requests.

- Remote to user (R2L): when attackers remotely try to gain access as an local machine user, by sending packets to that machine over a network and exploiting vulnerabilities.
- User to root (U2R): an attacker is able to access the system with a normal user account and then exploits system vulnerabilities to gain root access to control the system.
- Probing: an attacker scans networks, devices or applications to gather information and discover vulnerabilities.

These attacks aim to corrupt the privacy, non-repudiation, integrity and authenticity [Vac17] (see Section 2.2). An intruder can explore a lot of weaknesses on security systems, protocols, applications or settings, by using specific techniques and tools. Moreover, it can be performed based on social engineering [SCM09], where an attacker exploits a user who can grant access to the resource (a password or other information that compromises the security of the network and allows access to it), tricking them in order to reach the attacker goals [Vac17].

When using intrusion techniques, attackers exploit vulnerabilities in the implementation of systems, services, protocols, and others. There are also the problems generated by users and administrators, such as miss configuration and improper maintenance of the systems, inefficient passwords and outdated systems. Intrusion attacks are usually intended to steal or damage data [Vac17]. Some examples of attacks that may be performed on SDN are Port Scanning [AHR+10], FYN exploitation [GJW+14] and DoS [MZR14, AAG+15] attacks. Section 5.1 discuss what attacks can perform against SDN.

2.3.1 Port scan

Port scan or port scanning[2] is a very popular attack in SDN, mainly due to its characteristic of gathering the victim information. Typically, it is used to discovery some information about a target machine or system to perform another attack, *i.e.*, a port scan typically precedes another attack, which may be used to achieve a malicious goal [HPSR18].

Port scanning consists of sending several types of packets in order to know more about a target host or network. Through the answers obtained from analyzing these packets, the attacker is able to gather information that may help in future attacks. Some types of information that can be discovered include (not only): the activity of the servers, information regarding software used in the system, information about the firewall or network topology [NTL+18].

Due to their nature, scans can easily create a large number of different streams. According to Speroto *et al.*[SSS+10], there are three categories of scanning:

[2]We use port scan and port scanning as synonyms throughout this thesis

- Horizontal scan: when a source host scans a specific port on different target hosts.
- Vertical scan: when a source host scans several distinct ports of the same target host.
- Mixed scan: when there is a combination of vertical and horizontal scans.

Port scan attacks are typically performed by using specific tools, such as nmap [Lyo18]. Publicly available lists of potential targets, such as those provided on PasteBin[3] or gathered by services like Shodan[4] and registro.br[5] could also be used.

2.3.2 Denial of Service attacks

A common attack that can lead to system crashes is a DoS attack [MZR14], which can also be distributed (DDoS) using several different connections and machines [WCCL18]. This kind of attack aims to overload the target network, system or hardware on a way that it denies legitimate requests. Attackers can also use encryption on DoS attacks to bypass protection mechanisms and even to empower the attack. There are some different methods to do a DoS attack, such as Apache2, Back, Land, Mailbomb, SYN Flood, Ping of death, Process table, Smurf and Teardrop [PAGT07]. To perform a SYN flood attack, for example, a malicious user uses the action of opening a connection using the TCP protocol, in which a connection is opened just after finishing a three-way handshake. This handshake consists basically in the client sending a SYN package to the server; this package is used to notify the server that the client wishes to begin a connection; when the server receives this package, it answers to the client through a SYN-ACK package, meaning that the server received the connection request and it is waiting a confirmation from the client to establish the connection; the connection is fully initialized when the client receives the SYN-ACK from the server and answers to the server through an ACK. At this point the TCP connection is opened and both sides can exchange information [DDZX16] [MZR14].

A malicious user, when performing a SYN flood attack, sends SYN packages to the server, from several different computers, creating multiple requisitions. Once the server receives the SYN requests, it responses with a SYN-ACK and waits for the clients ACK. This waiting time is defined by the TCP protocol, and when the time out is reached, the connection is dropped. Once the malicious user knows that the server will wait some time before dropping the connections, this user will ask the server several new connections, and once the server answers with a SYN-ACK package, the attacker will drop this package and will ask for new connections. Hence, the server will be overloaded waiting to finish the

[3]pastebin.com
[4]https://www.shodan.io
[5]https://registro.br

connections, and the malicious user will consume the server resources, leading it to a DoS [YZ08]. Denial of service attacks are based on three main approaches [NPSR16]:

- Volumetric attacks: these are the most used attacks and are based on flooding a target with heavy traffic to exhaust its network bandwidth. The magnitude of this kind of attacks is measured in bytes per second.
- State exhaustion/protocol exploitation: these methods are used to exhaust the resources of devices by exploiting network protocols, in order to turn the network or operating system unavailable. Its magnitude is measured in packets per second.
- Application layer attack: this kind of method exploits the application layer protocols in order to crash the application or underlying a server. The magnitude of such attacks is measured in requests per second.

Figure 2.7 shows a short classification of the most common DoS attacks [NPSR16].

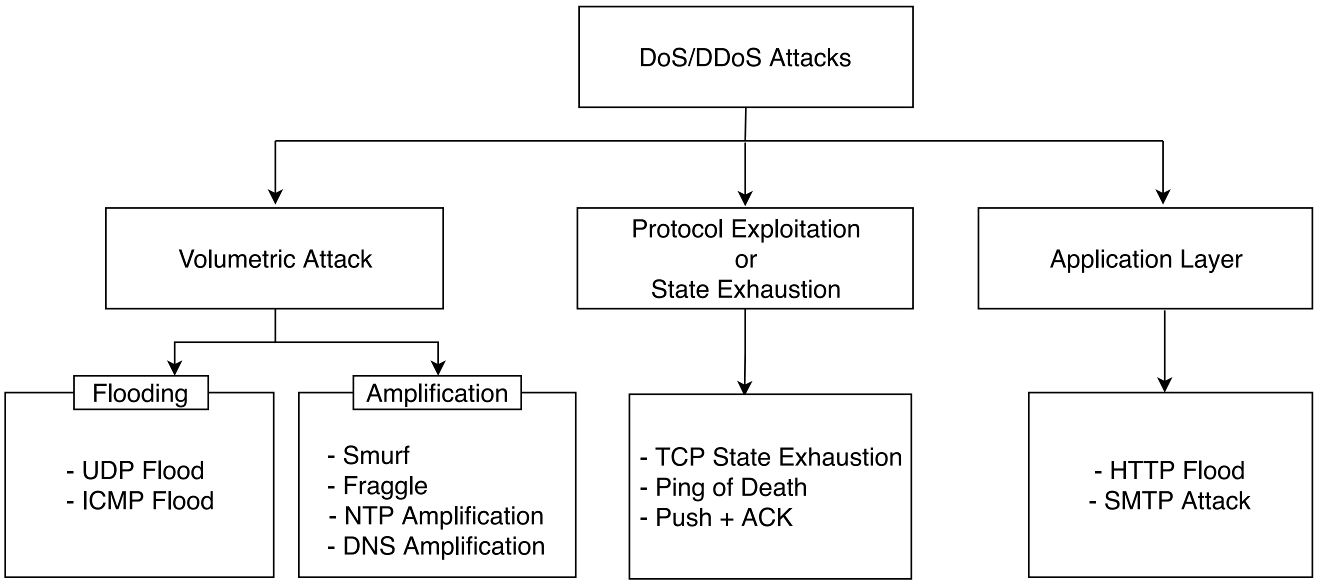

Figure 2.7 – DoS attacks

2.3.3 Insider attacks

Intrusion attacks can be classified according to their nature, motivated by insider and outsider threats[6] [WL06]. On one hand, outsider threats are generally outside the corporation (rivals, enemies or criminals) and they have limited opportunity to carry out their attacks. Outside attackers can only gain access by exploiting gaps or weaknesses in protection systems. On the other hand, insider threats have privileged access that enables them

[6]We use attacks and threats as synonyms throughout this thesis

to cause serious consequences, compared to outsiders. Normally, the access that enables insider attackers to cause so much damage is also essential to enable them to do their purpose. Moreover, the detection of such kind of malicious activity becomes even more difficult if encryption is used.

Usually, insider threats can be classified by unintentional threats and malicious threats. Unintentional threats are insiders who accidentally expose the organization data or the whole organization Information Technology (IT) infrastructure. Malicious threats are insiders who promote IT sabotage, theft of intellectual property or fraud [CMT12]. Malicious insiders can be involved in different activities, such as unauthorized extraction, data exfiltration, tampering with data or resources of an organization, destruction or deletion of critical data and assets, eavesdropping and packet sniffing with will intend and impersonation of other users via social engineering [SD16].

The internal attacks may not be result of a single problem, but of a set of small failures or vulnerabilities. Failures in safety procedures may allow users to find bugs that allow access to materials and tasks that they would not have authorization to. Incomplete or outdated documentation and poor access and permissions control can also contribute to insider attacks [Vac17]. Furthermore, network architectures and current systems are becoming more complex, making them even more vulnerable to this kind of attack, since they are more difficult to manage and therefore it is easier for the manager to “forget” to set some important security features in the network or system. One of the main motivations of inside attackers, is the sale of sensitive data, for example, in banking or e-commerce, for illicit enrichment [SD16] [WL06].

2.3.4 Other SDN threats

There are other attacks to an SDN infrastructure that could compromise an entire system. Among the different attacks, which in some way can compromise integrity, on one hand we can include: *Data Modification/Forging*, *Traffic Hijacking*, *ARP Poisoning*, *LLDP Spoofing* and *Lack of TLS Adoption*. On the other hand, attacks such as, *Controller Hijacking* and *TCAM Exhaustion* are focused on compromising the application availability [NPSR16]. A complete comparison on these attacks is shown in Table 2.4 [NPSR16].

Therefore, new network communication technologies must take care of security issues. An unsecured network channel can bring a lot of financial and operational disasters for a company. Privacy, integrity and authenticity are some of the most important concepts to preserve in a network environment. The efforts to improve network security are growing over the last years [KRV13]. IDS are examples of those systems that can help to protect computer networks and systems [NTL+18].

Table 2.4 – Categorization of attacks in SDN

Attack	**Affected security feature**	**Affects data plane**	**Affects control plane**	**Affects southbound interface**
Data modification/forging	Integrity	Yes	Yes	Yes
Traffic hijacking	Integrity	Yes	No	Yes
Controller hijacking	Availability	Yes	Yes	Yes
Denial of Service	Availability	Yes	Yes	Yes
Lack of TLS adaptation	Confidentiality, integrity	Yes	Yes	Yes
ARP poisoning	Integrity	Yes	No	No
LLDP spoofing	Integrity	Yes	No	No
Side channel attack	Confidentiality	Yes	Yes	No
TCAM exhaustion	Availability	Yes	No	No

2.4 Intrusion detection systems

One way to protect the network from attacks is to monitor traffic for malicious activities or policy violations. In order to monitor, identify, register and inform administrators of these systems and/or computer networks when some kind of malicious or suspicious activity occurs, Intrusion Detection Systems (IDS) [MdAN+11] are used. Those systems aim to identify attack attempts and typically report alerts to an administrator and generate logs that may be stored locally or centrally using, for example, a security information and event management (SIEM) [DRK+15] solution, which centralizes security logs from different protection tools, like Firewall, IDS and Antivirus.

Several IDS proposals have been developed in the past [MdAN+11] [PAGT07] [MZR14] [FNO+16] [EGdSSGSF16] [SSS+10]. These proposals can be classified according to several characteristics, such as type of analyzed data (logs) or packet data, type of connection (online or offline), or by the type of processing (centralized or distributed). However, the best-known classification models are signature-based and anomaly-based IDS [KID17]. Several detection methods for each model are found in the literature, and may be used individually or even combined to improve detection and protection [FNO+16].

- A signature-based or misuse-based IDS performs detection by comparing arriving packet data with packets stored in a database data. Those systems perform packet analysis by checking the payload, looking for a set of characters that identifies the attack. This character set is referred as the attack signature. However, packet inspection is difficult and even impossible to perform on networks with large bandwidth, *e.g.* several gigabits per second [GZL06] or on encrypted packets.
- An anomaly-based IDS, in turn, compares incoming data with a “normality model” that describes the normal behavior of the network. Significant changes on the behaviour of

flows, when compared with this model, are considered anomalies. For example, any application performing an attempt to unauthorized access to a system resource or a not expected traffic increase from a specific source or to a specific target. Examples of models to represent the network behavior can use neural networks, statistical analysis techniques, or probability theory. The main advantage of this type of detection is that it also detects attacks from previously unknown flows [OML10]. However, there may be instances where flows may be different from the expected normality but not necessarily malicious, resulting in false positive alarms.

An IDS can be classified according to the scope of its operation [NZOM16] [ONMZ16]:

- Host Intrusion Detection System (HIDS): in this mode, the IDS is responsible for monitoring the activity of a device, *i.e.*, seeks to identify attacks or suspicious events that compromise the security of this device, only analyzing the information transmitted over the network intended for this device.

- Network Intrusion Detection System (NIDS): In this scope, the IDS is responsible for analyzing activities involving all devices in the network. It analyzes the content transmitted on the network and attempts to identify malicious activities.

- Distributed Intrusion Detection System (DIDS): an IDS may also work in a distributed way. In this scope, several IDS instances run in the network to improve detection and performance.

In order to perform packet monitoring, the most appropriate solution is to use an IDS, which performs passive packets monitoring on the network. However, this type of analysis does not allow actions to be taken to prevent such attacks, as it only performs detection and generates alerts and logs. However, some studies [NTL+18] have improved IDS to act as a reactive solution that allows protection, *i.e.* to act as an Intrusion Prevention System (IPS).

2.4.1 IDS tools

Different IDS tools are used in other research, according to the literature [FNO+16] [Roe99] [dSJAF+15] [NZOM16]. Those tools can be used for identifying threats, as well as reacting to them. The most popular tools found in the literature, used specially for research purposes, are Snort, Suricata and Bro. Those tools can also be used for validating methods being developed by researchers, as they allow to customize rules and create new ones. Those rules can be enabled or disabled as required.

- SNORT: the most popular IDS tool found in the literature is SNORT [Com18]. It is able to perform real-time traffic analysis and packet logging on IP networks. It performs detection based on protocol analysis, content searching/matching and can also be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, DoS attacks, SMB probes and OS fingerprinting attempts, for example. Snort has a database with detection rules grouped according to each type of attack. Those rules can be activated according to the protection needs and can also be adapted and new rules can be written and inserted in the rules database. It is an open-source system and may be configured to work as a HIDS, NIDS and DDIS, as well as reacting on malicious activity detection, working as an IPS [Com18] [FNO$^+$16].
- Suricata: The Suricata IDS [Sur10] is a free and open-source NIDS. The project and code is owned and supported by the Open Information Security Foundation (OISF), whose goal is focused on security, usability and efficiency. Its engine can perform real time intrusion detection, inline intrusion prevention, network security monitoring (NSM) and offline files processing. Suricata provides powerful and extensive rules and signature language for network traffic analysis, and has powerful Lua scripting support for complex threats detection [Sur10]. It uses standard input and output formats like YAML and JSON integrated with tools like existing SIEMs [DRK$^+$15] and others.
- Bro: The Bro IDS project [Pax98] [Bro] has been developed by Vern Paxson, who currently still leads the project in cooperation with researchers and developers at the International Computer Science Institute in Berkeley and the National Center for Supercomputing Applications in Urbana-Champaign. It is an open-source system and provides a comprehensive platform for more general network traffic analysis. Its user community includes universities, research labs, supercomputing centers, and open-science communities [Bro].

2.4.2 IDS data sets

Several data sets are available for evaluating attacks patterns and to validate detection methods. Some of the most used data sets in the literature are described next. Those data sets were captured or generated over a realistic network configuration and provide labeled data. However, to the best of our knowledge, there is no IDS data set for SDN in the literature.

- ISCX 2012: The Canadian Institute for Cybersecurity at the University of New Brunswick [UNBb] has build several IDS data sets that are available for research purposes. One such data set is the UNB ISCX IDS 2012 Intrusion Detection Data set [SSTG12] [UNBc]. Real traces are analyzed to create profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP. Labeled network traces, with full packet payloads in pcap format are also included. This data set was generated by seven days of network activity (normal and malicious), grouped in files that have follow activities: a) Normal Activity only; b) Infiltrating the network from inside and Normal Activity; c) HTTP DoS and Normal Activity; d) DDoS using an IRC Botnet; e)Brute Force SSH and Normal Activity.

- CICIDS2017: The CICIDS2017 [SLG18] data set was also developed by the Canadian Institute for Cybersecurity in the University of New Brunswick [UNBb] and contains benign and the most up-to-date common attacks, which resembles the true real-world data.The CICIDS2017 data set is composed of labeled network flows, with full packet payloads in pcap format, the corresponding profiles and the labeled flows and Comma-Separated Values (CSV) files for machine and deep learning purposes. Victim and attacker networks information are also available. Some results of the network traffic analysis using CICFlowMeter [UNBa] with labeled flows based on the time stamp, source and destination IPs, source and destination ports, protocols and attack (CSV files) are included and available. CICFlowMeter is a tool [UNBa] to provide flow analysis on network traffic, and generates CVS files with features as need, being specially useful for machine learning and data mining methods.

- KDD Cup 99: The KDD Cup99 [DAR] is an IDS data set based on DARPA/MIT Lincoln Lab original data sets [MMIoT] and represents the activities at USA Air Force LAN, including normal traffic and malicious activities that were injected. It has 4GB of compressed TCPDUMP data of seven weeks network traffic, representing five million connection records with 100 bytes each. In this data set, DoS, U2R, R2L and Probing attacks [BZ17] were injected. It is an old data set, however, it is still usefully for detection methods evaluation. A list of features and attacks is provided, as well as labeled and unlabeled data.

2.5 Encrypted traffic classification and analysis

Nowadays, different applications generate several traffic types with different service and management requirements in modern networks. Thus, traffic classification plays a key role in network management for identifying the data type, communication protocol, detecting malware and to improve quality of service on traffic prioritization or restriction. Several network traffic classification methods are available in the literature, which are usually based on the knowledge of packet structure or communication patterns [VD15].

Initially, methods are distinguished according to classification input, classification technique and classification output. The classification input is related to traffic characteristics, such as hosts, packets and flows. Table 2.5 shows the main characteristics that can be used as classification input.

Table 2.5 – Classification input level

CLASSIFICATION INPUT		
Traffic payload		Use of application data
Traffic properties	Host community	Graph metrics (diameter, connection degree)
	Host	Number of connections, opened ports
	Flow	Flow size, flow duration
	Packet	Packet sizes, inter-arrival times
Hybrid and miscellaneous		Combination of inputs, external knowledge

The classification technique defines the method used for classification, such as payload inspection (using, for example, Deep Packet Inspection (DPI)), machine learning and statistical-based methods. Table 2.6 shows the main techniques used for traffic classification [VD15].

Table 2.6 – Classification input level

CLASSIFICATION TECHNIQUE		
Payload inspection		DPI, examination of first N bytes
Graphical techniques	Graphlets	Relationship between features (ports, addresses)
	Motifs	Patterns of communication
	Social networks	Graph of communication
Statistical method	Basic statistical	Probability density functions of, e.g. packet sizes
	Heuristics	Port-based classification
	Profiles	Host profiling, usage of packet sizes and direction
Machine learning algorithm	Supervised	Hidden Markov models, naive Bayes, k-nearest neighbour, support vector machine
	Non-supervised	Clustering of unlabeled traffic, k-nearest neighbour
	Semi-supervised	Clustering of mixed traffic, k-nearest neighbour
	Reinforcement learning	__
Hybrid and miscellaneous		Combination of methods, external knowledge

The classification output identifies how traffic objects, such as packets and flows, can be mapped to traffic classes, like application types or protocols. These classes may have different granularity according to the ISO/OSI reference model, being, for example, application protocols such as HTTP or even be more fine grained to identify the application that generates the data, such as a Facebook chat. Table 2.7 shows some output traffic objects and classes that classification methods can generate [VD15].

Accurate traffic classification methods that can identify the encrypted traffic type, such as communication protocol, are a core part in analyzing network encrypted traffic. For example, to identity TLS packets, methods based on the knowledge of the packet structure, especially the unencrypted packet parts such as content type, version and length can be used. Moreover, it can also be identified by analyzing its behaviour, such as the knowledge

Table 2.7 – Classification output level

CLASSIFICATION OUTPUT	
Traffic objects	
Host community	Host community is assigned a class, e.g. community of HTTP serves
Host	Host is assigned a class
Flow	Flow is assigned a class
Packets	Packet is assigned a class
Traffic classes	
Traffic cluster	Bulk or small transactions
Application type	Game, browsing, chat
Application protocol	HTTP, HTTPS, FTP
Application software	Client software such as Mail client, FTP client or web browser
Fine grained	Skype voice call, Google search, Facebook chat
Anomaly	Port scan, brute-force attack
Hybrid and miscellaneous	Combination of outputs or classes, external knowledge

of a number and approximate size of packets sent during the unencrypted initialization phase [VD15]. Several tools that can be used for network traffic classification and also to distinguish specific network applications are available. Typically, payload inspection techniques are used on the traffic payload classification input to map flows to application protocols by those tools. Examples of classification tools are:

- PACE [CEBBR14]: is a commercial classification library written in C that is able to classify several protocols and applications, such as BitTorrent or Skype traffic. It uses pattern matching augmented by heuristics, behavioural and statistical analyses.

- Cisco Network Based Application Recognition (NBAR) [CEBBR14]: it is also a commercial tool that is primarily used on Cisco routers to provide quality of service and security on the network. This tool is able to recognize stateful protocols, non-TCP and non-UDP protocols.

- nDPI [CEBBR14]: this is an open-source classifier derived from PACE. It analyses at most eight packets from each connection for classifying traffic, but each packet is examined individually. nDPI has only an SSL decoder that extracts the host name from the server certificate for encrypted traffic identification. Thus, nDPI is able to identify specific network applications using these names.

- Libprotoident [AN12] [CEBBR14]: this is an open-source C library that inspects only the first four bytes of a packet payload for each direction, being faster than others, but also a less accurate method for traffic classification. A combined approach of pattern matching, payload size, port numbers and IP matching is used in the classification process [ICFCG18].

2.6 Chapter summary

This chapter introduced the key concepts associated with SDN, SDN security and threats, IDS/IPS methods and datasets, cryptography and encrypted traffic management. Initially, the SDN paradigm is described. We showed that the SDN architecture is composed by three main layers, which are the Application layer, Control plane (also know as Control layer) and the Data Plane (also known as Infrastructure layer). Three communication interfaces are also provided in SDN, being a Northbound, which provides communication between the controller and its applications, Eastbound/Westbound, which provides communication among other controllers, and Southbound interfaces, which provide communication between the controller and the Data plane (e.g. SDN switches). This chapter also described the main SDN protocols, being OpenFlow the most used nowadays. Then a discussion about the SDN controller and a short comparison of some controllers is presented. A brief description about cryptography is made in Section 2.2. Next, several attacks are described on a generic form, being included attacking methods found against traditional networks. This is useful to understand attacking methods and what impact they have, which is important to analyze security in SDN and also to design protection methods. The described attacks could also be made using encrypted data, improving their malicious purpose because the encryption and decryption process increases the processing power consumption, degrade performance, limiting capacity and introducing latency in the network. Moreover, such data could bypass protection systems, as such systems typically can not analyze encrypted data.

3. SYSTEMATIC MAPPING STUDY

Researchers across the world have been working to improve IDS. On one hand, a lot of methods and systems have been proposed, described, implemented and tested. On the other hand, limitations, vulnerabilities and new issues still remain. Hence, it is necessary to improve our expertise and skills related to network security, specially NIDS, before we further explore solutions in this area. Therefore, a Systematic Mapping Study (SMS) was executed on network intrusion detection methods. This allowed us to identify the state of the art and challenges related to NIDS, their vulnerabilities and security mechanisms. Based on that, this thesis scope was defined.

The method used by the SMS was based in processes used by similar studies conducted in Computer and Network Security (e.g. Bertoglio *et al.* [BZ17] and Software Engineering (e.g. Peterson *et al.* [PR06]). Thus, a set of related papers available at ACM Digital Library[1], IEEEXplore[2], Compendex[3] and Scopus[4] databases were analyzed and classified. Those papers were selected through a search string obtained by specific research questions established to this work. Figure 3.1 shows the main steps of our SMS.

3.1 Defining scope

The main goal of this SMS is to map the state of art on NIDS based on the published research that is available in the main digital libraries. This process is helpful to identify which are the main attacks, map the main methods used for network intrusion detection, their limitations and open issues. It can be considered as a first step as the research for this thesis has started.

3.2 Establishing research questions

Research questions (RQ) can be used as a guide to help researchers to define research scopes. In this SMS, a core goal was to made a research about how network protection systems act against malicious attempts and know how researchers are improving them. Those questions can also be used to define a search string that can be used on the search engines to retrieve related papers in a systematic way. Thus, the definition of the research questions plays a core role in this work, and follow questions were established:

[1]https://dl.acm.org
[2]https://ieeexplore.ieee.org
[3]https://www.elsevier.com/solutions/engineering-village/content/compendex
[4]https://www.scopus.com

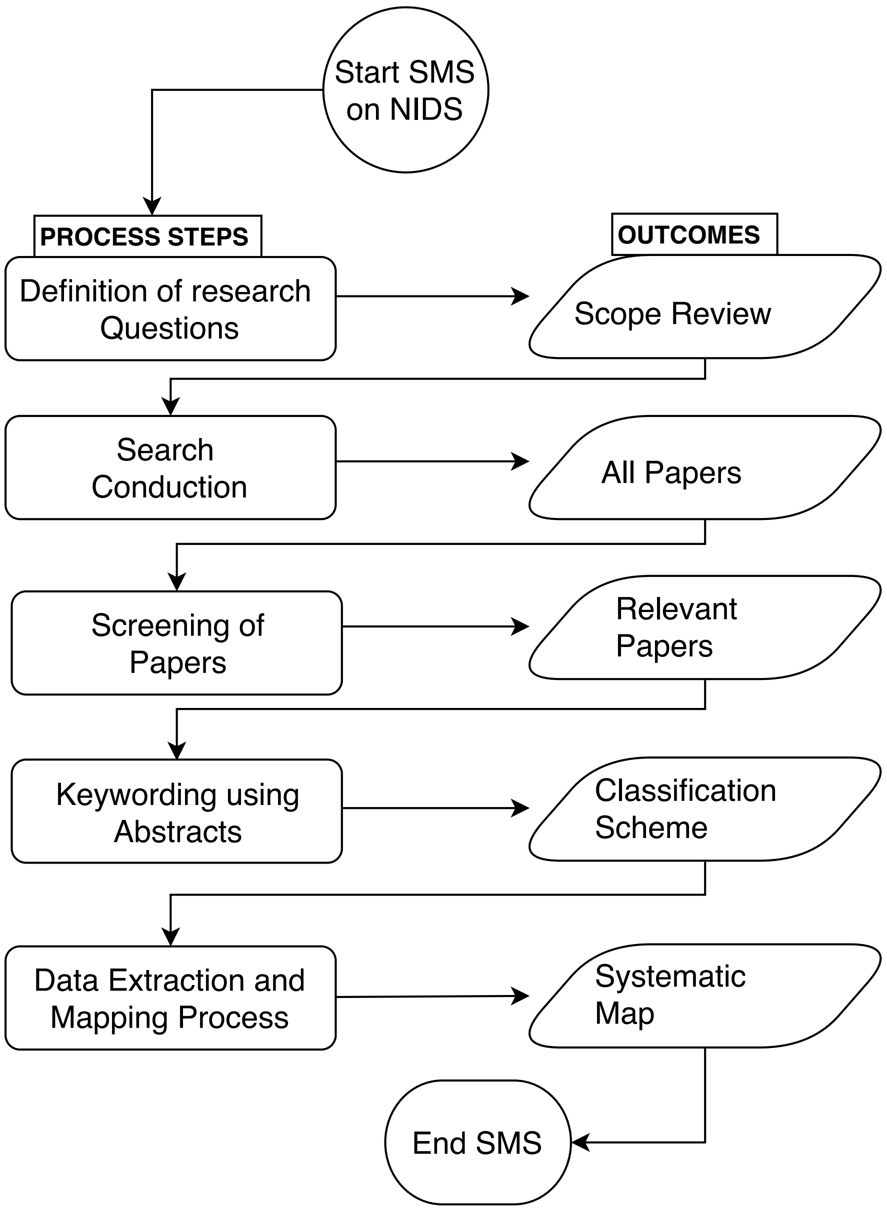

Figure 3.1 – Systematic Mapping Process

- RQ1: Is NIDS a current relevant topic for research?
- RQ2: What techniques and methods have been proposed to improve NIDS?
- RQ3: What are the main NIDS tools?
- RQ4: What are the main limitations of NIDS?

3.3 Inclusion and exclusion criteria

The definition of the Inclusion Criteria (IC) and Exclusion Criteria (EC) are used to select only works that are in the research scope, being useful to include all the works publicized and also to reduce the number of papers that are out of scope returned by the search engines through the search string. Thus, if a paper is classified in only one IC, it will be included as a primary study. However, if a paper is associated to any EC, it will be excluded. In this SMS the following IC and EC were defined:

- IC1: The primary study must propose an approach, technique, methodology, method or tool for network intrusion detection.
- IC2: The primary study must address the use of NIDS approach, technique, methodology, method or tool to support one or more NIDS tasks.
- IC3: The primary study must propose a solution to prevent or to detect a specific network intrusion method (attack);
- EC1: The primary study is not published from 1999 to 2015;
- EC2: The primary study is written in a different language than English.
- EC3: Studies that do not contain some type of evaluation, such as case study, experiment, or proof of correctness. This evaluation must contain some kind of analysis that shows the achieved results.
- EC4: Studies that present some NIDS approach, technique, methodology, method, tool or framework, but do not provide sufficient information about its implementation or evaluation;
- EC5: Studies that present some NIDS intended to hardware, like FPGA;
- EC6: Papers that are not completely available online;

3.4 Research strategy and search string

The key words of the research questions where extracted and used together with the logical operators “AND” and “OR” to define a search string. This string was applied on the chosen search database engines to retrieve the papers related to our research. It is important to note that this string must be adapted according to the search engine.

This search string was applied only in engines that (1) have a web-based search engine; (2) have a search mechanism that allows the usage of keywords; and, (3) contain computer science papers. Our selection includes the ACM Digital Library, IEEEXplore, Compendex and SCOPUS.

Logical operators were used to define the search string in order to apply the defined terms for search the papers. “OR” operators were used to establish the relationship between synonyms and similar terms, and “AND” operators were used to connect population, intervention and outcome terms. The entire terms and their acronyms were used. Besides, structured questions were used to compose the search string. This question structure is show in Table 3.1.

- Population: published research papers on NIDS.
- Intervention: methods to identify network intrusion.
- Outcome: the expected results are the approaches in which network intrusion methods are applied to from 1999 to 2015.

Table 3.1 – Definition of the search string

STRUCTURE	TERMS	SYNONYMS
POPULATION	Network security Intrusion detection Network Intrusion~ Network-based intrusion	IDS, NIDS, IPS, NIPS,HIDS
OUTCOMES	Approach	Method, Technique, Methodology, Detection, Identification, Prevention

To extract the relevant data from the selected papers, we produced a form that would help to the documentation and to organize the studies to this work. The following data was extracted for each study:

- Database: ACM Digital LLibrary, Compendex, IEEExplore and SCOPUS.
- Source: full reference conference, book, journal name
- Title
- Abstract
- Authors
- Year
- Publication Type: Book Chapter, Conference, Journal, Symposium, Workshop, or Other
- Document Type: Article, Collection, Proceeding, Periodical, Technical Report, or Thesis.

After executing the search string in the aforementioned search engines, a table was constructed with those information about the papers. Some databases provide a tool to export the results from the search query to a comma-separated values (CVS) file. This tool was very helpful to form the table with the papers list. Other databases do not have this tool, and in this case it is necessary an extra effort to develop one or to construct the list manually.

The next step taken was the applying of the insertion and exclusion criteria over the retrieved studies. To fit these studies in the proposed criteria, information like publication year, paper title and abstract were read and analyzed. Additionally, similar or redundant studies were discarded by selecting only the most recent ones. The output of this process is the set of primary studies that are going to be addressed by this work. To further broaden the range of material used by this work, relevant studies referenced by the primary studies were also verified and, when appropriate, selected to be used in the systematic mapping as well. The same criteria was applied over these referenced studies.

3.5 Selection process

As a structured work, this systematic mapping has been followed some steps. Our selection process is divided in six steps:

- Step 1. Search databases: initially, strings were generated by means of the selected keywords and synonyms. Initial selection: an initial selection was carried out based on the defined IC and EC.
- Step 2. Eliminate redundancies: there were some redundancies since some studies were returned by different search engines. In this step, those redundancies were eliminated and recorded.
- Step 3. Intermediate selection: the title and the abstract (reading the introduction and conclusion when necessary) were read for each study returned by the search engines.
- Step 4. Final selection: In this step, all studies were completely read, and the same criteria as in the intermediate step were applied.
- Step 5. Quality assessment: Based on the quality criteria, we evaluated the quality of studies that were read in the final selection step. The quality criteria were evaluated by two researchers, individually.

3.6 Results

This section describes the main results of this SMS. The search string was used for querying four search engines (IEEExplore, SCOPUS, Compendex and ACM Library). Together, the four engines returned a total of 1024 publications (including duplicates). After inclusion and exclusion criteria were applied, 238 studies were selected to be read completely. To apply the criteria and select appropriate publications, the paper's title and the abstract were read and analyzed. Table 3.2 shows the number of publications retrieved and selected according to each search engine.

Table 3.2 – Found and selected studies by each search engine

SEARCH ENGINE	**PAPERS RETRIEVED**	**PAPERS SELECTED**
IEEEXplore	825	179
ACM Digital Library	215	61
Scopus	18	9
Compendex	27	12

By reading the entire papers selected, it was observed that some of them are intended to embedded systems, such as FPGA or SCADA networks. Others were written in other languages than English, Portuguese or German. Some studies also have publicized only the abstract. All those papers were excluded from the selection, resulting on 177 papers that meet to the scope defined to this work (only high level software-based NIDS).

After the selected papers were read, the research questions could be answered, as follow:

RQ 1: Is NIDS a current relevant topic for research? A key question to define a research topic, is analyze if it remains a current research topic. A good approach to identify it, is analyzing the number of works that were published on the last years. In this SMS, we grouped the selected papers according to the publication year and could observe an increasing number of works being developed in the last seventeen years.

RQ 2: What techniques and methods have been proposed to improve NIDS? To answer this question, selected papers were also grouped according to their purpose, being:

- Framework: a NIDS framework is proposed/described.
- Survey: authors present a discussion off current NIDS.
- New method design, implementation and evaluation with results.
- New method proposal, being only a theoretical description. Evaluation of existing methods. IDS datasets evaluation or proposal or description.
- Approach/Framework for NIDS simulation and evaluation.

Methods based on artificial intelligence and machine learning are widely used to develop new detection methods based in anomaly-detection. Hybrid approaches, applying misuse and anomalies are also used in several works. Patterns are also used to define a standard traffic model for the network, considering their systems and users, being presented methods that are able to detect deviations from this standard to identify attacks.

Another interesting technique is the use of multivariate analysis, such as Principal Components Analyses (PCA). These works are intended to protect specif networks, being Wireless, Smart grids, cognitive radio, virtualization, mobile networks, cloud, sensors, SDN, traditional corporate IP networks, real time monitoring and web server.

RQ 3: What are the main NIDS tools? Several selected papers describe a variety of NIDS methods and algorithms. Some of them, can be used by itself as a NIDS, however, others should be integrated as a complement or as a rule (s) on a NIDS. The most standard tools found in the studies were Snort, Bro and Suricata.

RQ 4: What are the main limitations of NIDS? The encrypted data analysis remains as a limitation of the current NIDS, as only a few works that present an approach or method to analyze such traffic were found. Another limitation found, is the inability of the methods to analyze all the packets in high-speed networks due to time requirements. The high number of false positive alerts present in several works also is still a limitation for adopting such methods in real networks. Thus, overhead, performance and high false positive alarms are core issues to be solved in NIDS. For methods based on misuse, it may be a lag between the time that a new attack is discovered and a new signature is created and applied to the NIDS. Moreover, some attacks are geared for specific versions of software that are usually outdated. By updating the signatures in order to consider current versions off this software, signatures intended to the old versions may be changed or replaced by new ones. Thus, outdated signature databases can turn the NIDS vulnerable if those softwares are not updated to current versions. Noise on the network channel can also limit the IDS effectiveness. This noise can consist of bad packets generated by software bugs and corrupt DNS data, for example, can induce the IDS to generate a significantly high false-alarm rate, the so called, false positive. Sometimes the number of real attacks is far below the number of false-alarms, and therefore this real attacks are often missed and ignored. Finally, it is also important to mention evasion techniques used by intruders to bypass the IDS, such as pattern change evasion, address spoofing, avoiding defaults, fragmentation and coordinated low-bandwidth attacks.

3.7 Discussion

The results show that the research community is making massive efforts to solve problems related to NIDS. However, intruders and attackers are improving their methods and challenges are still remaining with open issues. The most core challenges found are:

- Currently, NIDS are not able to analyze all packets and flows in high-speed networks, when accuracy and performance are required. Usually, the methods developed for NIDS found in the literature when used in real environments are a kind of bottleneck on the network and therefore are not appropriate to be used, due to the overhead that they inject to the network or system, high false positive rate and performance impact.

- The usage of cryptography is increasing nowadays and represents another big challenge for current NIDS, as they are usually not able to analyze such traffic. New network technologies, such as SDN, are arising as a new paradigm, and could be used to help on improving communication. However, security must be considered to allow the usage of such technologies in real environments. Thus, SDN offers a lot of research opportunities related to security.

3.8 Chapter summary

This chapter described the SMS performed for this thesis. The SMS output shows that intrusion detection methods development remains a relevant topic nowadays, as an increasing number of works was found. It also shows that the main limitation of current IDS, found in the literature, is their inability to analyze encrypted data packets or encrypted flows, and only a few works that address IDS for SDN were found. SDN may be adopted on several applications, such as IoT (Internet of Things) [BAP+16] and Industry 4.0 [Fouf] devices. Therefore, it is important to improve security in SDN, specially when encryption is used, as the amount of encrypted traffic is increasing in the Internet and also in internal networks due to aplications security needs.

4. RELATED WORK

In this chapter, we briefly describe some essential concepts that underlie our research. The purpose of this chapter is to provide an overview of the state-of-the-art regarding methods for detecting attacks that can be performed using encrypted data and also methods for traffic (encrypted and not encrypted) classification. To the best of our knowledge, there is little work describing encrypted traffic analysis, classification and encrypted malicious traffic detection, specially for SDN. Therefore, works intended to traditional IP networks are also included, as those approaches may be helpful for our research and as there are only a few works that are focusing in SDN.

4.1 Port scan detection approaches

As a first step to perform a specific attack, usually a port scan is performed. By lunching a port scan, attackers can discover some information, such as open ports and applications that are running, about a target network or system that can be used on a specific attack, such as a denial of service.

- The evaluation of different behaviors in an SDN environment is becoming a promising approach to identify attacks. Ali *et al.* [AHR+10], for example, proposed an algorithm based on packet content analysis to improve the anomaly detection and, thus, to avoid an attack to an SDN infrastructure. Their proposal consists of a Progressive Security-Aware Packet Sampling (PSAS) algorithm that enables an online inline anomaly detector to achieve higher accuracy by sampling larger volumes of malicious traffic than random sampling, without increasing the detection cost. The dataset to evaluate the algorithm used traffic from a port scan and DoS attacks. For each analyzed packet, a score that represents if the packet is malicious or not is given. That score is reanalyzed in each switch that the packet traverses. The packet is then considered malicious depending on the score it achieves each time it is analyzed.
- Lagraa and François [LF17] proposed a solution to detect and to discover patterns of port scans. In order to analyze a large amount of data, they proposed a model to convert TCP packets into a graph containing packet information and their relations. This information is grouped by the data receiving port. Although the results presented are promising, performance and hardware requirements were not discussed.
- Considering overhead to an SDN controller, Zhang *et al.* [ZGYW16] proposed a solution to mitigate port scans in SDN networks using a technique called Port Hoping. Their approach consists of dynamically mapping ports to unused ports. Consequently, it can

confuse potential attackers and increase the attack cost. Although measured overhead was low (increase of 2.1%-4.9% on CPU usage), this approach did not stopped attacks. Consequently, attackers could be able to continue the attacks using other ports.

4.2 DoS detection approaches

Denial of service attacks are still evolving. Therefore, several researchers are addressing methods to detect and prevent such attacks in traditional networks and also in SDN.

- Boite *et al.* [BNR+17] also proposed an IPS solution to detect DoS on SDN infrastructures. Their approach consists of an anomaly-detection IPS that uses an entropy-based algorithm. To reduce overhead in the detection and prevention, their algorithm relies on three main stages required to handle the security management function: monitoring, detection and mitigation. Once a violation is detected in the monitoring stage, new flow rules are installed into the switch with a high priority, hence new suspicious packets will be discarded immediately. However, performance costs to implement their solution in an SDN controller were not discussed.
- Jazi *et al.* [JGSG17] present a detection method of Http-based application layer DoS on web servers. This approach is based on nonparametric Cumulative Sum (CUSUM) algorithm. They goal is to analyze the impact of sampling techniques on detection of application layer DoS attack, as in production environments detection is commonly performed on a sampled subset of network traffic. Real traffic traces of application level DoS attacks were used to validate the effectiveness of this approach. The results also show that the most of sampling techniques developed specifically for intrusion detection domain introduce distortion in the traffic that could impact detection.
- Osanaiye *et al.* propose a change-Point cloud DoS detection method using packet inter-arrival time. Their approach is based on a change-point monitoring algorithm that detects DoS flooding attacks against cloud services by examining the packet inter-arrival time. This method uses both Flow-Based Classifier and the CUSUM algorithm to detect cloud DoS flooding attacks, based on the packet inter-arrival time rate. They argue that most DoS attacks are automated and thus usually have similar patterns that can be distinguished from normal traffic patterns, and can therefore be tracked using a CUSUM algorithm. Trace-driven simulation and empirical evaluation to detect changes within traffic flows that have both normal and attack packets were conduced to validate the proposed solution. Moreover, flooding attacks from CAIDA DoS benchmark dataset and normal traffic trace from Auckland-VIII were used to validate the approach. Results

are indicating that optimal performance was achieved by CUSUM during 100 and 150 packet counts for normal flow to detect an equal amount of attack packets [OCD16].

- A One-Class NIDS for SDN-Based SCADA Systems is proposed by Silva *et al.* [EGdSSGSF16]. The authors discuss the usage of SDN in the deployment of next generation SCADA systems and present a NIDS for SDN-based SCADA systems. In order to build this NIDS, SDN is used to extract network information and is for monitoring the communication between power grid components. Statistics are collected from network devices periodically, which are then processed by One-Class Classification algorithms. A DoS attack was simulated and an accuracy of 98% was achieved in the results. Thus, the authors argue that their approach can be effectively used to detect cyber-attacks targeted against SCADA systems.

4.3 Generic encrypted attacks detection

Encrypted threats detection is a current topic that researchers are working on. Follow, some research that aims to detect malware in encrypted traffic are presented.

- An approach to identify the most appropriate features and machine learning algorithms that should be used on encrypted malware traffic classification is presented by Anderson *et al.* [AM17]. The authors conclude, based on experimental evaluations, that the random forest is the most effective algorithm in such task. their analysis is based on millions of TLS encrypted sessions collected over 12 months from a commercial malware sandbox and two geographically distinct, large enterprise networks.

- Anderson *et al.* [APM17] present a research intended for detecting hidden malware in TLS traffic. The authors discuss that TLS introduces a complex set of observable data features that allow inferences about the client and the server. Based on experimental evaluations, they show that these features can be used to detect and understand malware communication and behavior without data decryption, preserving the privacy of the benign uses of encryption. A detailed study of TLS usage by malware and enterprise applications is also presented. The authors provide a general analysis on millions of TLS encrypted flows, and a targeted study on 18 malware families composed of thousands of unique malware samples and tens-of-thousands of malicious TLS flows. Finally, the research show that the performance of a malware classifier can be correlated with a malware family's use of TLS, i.e., malware families that actively evolve their use of cryptography can be more difficult to classify, and the malware's usage of TLS is distinct in an enterprise setting. These differences can be used in rules and machine learning classifiers to develop protection methods.

4.4 SDN attacks identification

As a new paradigm in computer networks, SDN also have several vulnerabilities that may be used by attackers for malicious purposes. In the literature, some research are addressing such vulnerabilities and are proposing protection methods. Although the majority of those methods do not address encrypted attacks, their approaches are important to understand existent protection and to evaluate and extend them in order to increase security in SDN against malicious encrypted traffic.

- A Predicting network attack patterns in SDN using machine learning approach is presented by Nanda *et al.* [NZD+16]. In this work, the authors proposed machine learning algorithm to predict the vulnerable host in SDN that may probably be attacked. C4.5, Bayesian Network, Decision Table and Naive-Bayes learning algorithms are used to predict the host that will be attacked based on the historical data. Based on results, they argue that the use of machine learning algorithms to accurately predicting the potential vulnerable host can help in defining new security rules for SDN controller so that the attacks do not materialize. Experimental results show an accuracy of 91.68% by using Bayesian Network, indicating that from the total 278,598 attacks generated on the experiments, Bayesian network was able to accurately predict 254,834 attacks [NZD+16].
- As DoS attack against controllers is one of the key security threats of SDN, a detection method for a novel DoS attack against SDN controllers by vast new low-traffic flows is presented by Dong *et al.* [DDZX16]. They proposed method is designed to detect the DoS attack and to further locate the compromised interfaces the malicious attackers have connected to. Initially, the flow events associated with an interface are classified, then Sequential Probability Ratio Test is used to make decisions, which has bounded false negative and false positive error rates. The DARPA Intrusion Detection Data Sets are used to evaluate effectiveness and performance of the proposed methods. A discussion is made and a comparison of the methods being proposed to three other detection methods is presented, based on the percentage, count, and entropy of the flows, respectively, showing that the developed methods are more effective in terms of promptness, versatility and accuracy [DDZX16].
- Seeber *et al.* [SSR15] present towards an SDN-enabled IDS environment, an approach for redirecting suspicious traffic taking advantage of properties of OpenFlow in SDN. The proposed approach uses the OpenFlow switch flow table rules set as a light IDS that maintains a history of recurring events including involved IP addresses as well as information from external sources, like public available black-lists, white-lists, geolocation data and their severities. They method performs correlation of IP-to-location

mappings to support the suspiciousness value of an event and additional consumer triggered event lists. This value is computed based on the Common Vulnerability Scoring System, including how unlikely a vulnerability is exploited and the source of an attack based on the IP address. Thus, suspicious traffic is redirected to various IDS for further inspection. The proposed method is able to drop bogus traffic and forward DoS suspicious traffic to a specific DoS detection system. In addition, this system maintains an attack history directly in the steering entity enhanced by existing up-to-date blacklists and in a geolocation database. Results are presented as a Proof of Concept in a laboratory environment, where real traffic traces were used [SSR15].

- *Paladi* [Pal15] describes Towards Secure SDN Policy Management, reviewing the state-of-the-art for network policy verification for SDN deployments, identify existing challenges and outline a secure framework for network policy management in SDN deployments, aiming to contribute towards creating secure and trusted cloud deployments. The need for a comprehensive approach towards SDN policy management is outlined, including generation, verification and enforcement of network policies. An overview of the existing solutions is present and a set of challenges that are still found in SDN are discussed.

- Giotis *et al.* [GAA+14] demonstrated that OpenFlow statistics collection and processing may overload the control plane. Thus, they proposed an architecture responsible for handling separately the data collection process from data flow monitoring. The main attacks that were evaluated are DoS and Worm propagation. Port scan was also considered, since it is usually performed prior to an attack. The attacks are detected through system entropy change analysis. Their proposal consists of two algorithm modules implemented on the NOX controller [SFF+13], one that is responsible to identify the anomaly and another to execute a mitigation task. However, their proposal does not discuss whether their analysis may be applied to ciphered packets or not.

4.5 Traffic classification

Traffic classification methods are fundamental in network management. It is important to identify the traffic in order to better define quality of service rules, predict future traffic matrices and demands and also to detect anomalies in traffic behavior. Several methods intended to traffic classification were found, however, only a few are addressing SDN flows. Follow we briefly describe those methods.

- An identification and selection of flow features for accurate traffic classification in SDN is presented by Silva *et al.* [dSMB+15]. An architecture to collect, extend and select flow features that can be used for traffic classification in OpenFlow-based networks is described to provide an extensive set of flow features that can be analyzed and refined and to be capable of finding the optimal subset of features to classify different types of traffic flows. Features are grouped as Statistical features, Scalar features and Complex features. Evaluations are made in three different scenarios: DoS, FTP and video Streaming. Their results show that when the optimal subset of features is used in a PCA, the classification accuracy is 97.33% in a DoS scenario, 94% in FTP and 88.67 in a video streaming scenario.

- A tool support for the evaluation of anomaly traffic classification for network resilience is presented by Silva *et al.* [dSJAF+15]. The authors discuss that traffic classification can help in malicious traffic detection, as well as for handling the traffic according to its class. Two algorithms for anomaly traffic classification based on machine learning were implemented and analyzed to provide a toolset for the simulation and analysis of anomaly traffic classification algorithms. This is helpful to identify the best algorithms, configuration parameters and network policies, when different types of attacks and anomalies are simulated. This toolset also supports the analysis and comparison of classification techniques. In this work, the authors have extend the PReSET [dSJAF+15] toolset to be used for the investigation, comparison and analysis of anomaly traffic classification algorithms based on machine learning. Evaluations were made in a scenario that has DoS attacks.

- ATLANTIC is a another framework for Anomaly Traffic Detection, Classification, and Mitigation in SDN developed by Silva *et al.* [dSJAFG16]. The authors discuss that SDN provides propitious environments for anomaly classification schemes and show that ATLANTIC can be used for categorizing traffic anomalies and using the information collected to handle traffic profiles as required, such as blocking malicious flows. The proposed framework uses information theory to calculate deviations in the entropy of flow tables and then uses machine learning algorithms to classify traffic flows. Port scan and DoS attacks were simulated to validate the proposed framework.

4.6 Encrypted traffic classification

As an increasing number of applications are using encryption on data communication, encrypted traffic classification becomes also a core task in current networks management, improving traffic visibility, specially to better apply quality of service rules and also to detect anomalies. In this section, some works related to such methods are presented.

- A novel algorithm for encrypted traffic classification based on sliding windows flow's first N packets is described by Liu *et al.* [LCCY17]. This method is intended for encrypted traffic classification, specially SSH and P2P traffic, based on sliding windows and only considers first N packets of each flow classification algorithm. Thus, they could reduce the flow characteristics feature dimension, as well as the number of packets in each traffic flow. The classification algorithm presented, called WFNP, selects flow statistics feature. After, they use C4.5 classification method to classify the encrypted traffic. Experimental results, based on a dataset that was generated by the authors, show that under a reduced dimension of encrypted traffic flow characteristics and also a reduced number of each flow data packets, average classification accuracy is around 95%.

- McGaughey *et al.* [MSSK18] presented a work with a systematic approach of feature selection for encrypted network traffic classification. This approach is based on a statistical analysis technique and uses the fast orthogonal search (FOS) algorithm to select a subset of features that allows to distinguish it from a large set of features derived from the data. After, a k-nearest neighbor (kNN) classifier is then applied to classify the network traffic using the features selected by FOS. The FOS algorithm selected a subset of 12 features from a set of 2,839 original ones. They conclude that the kNN classifier using these 12 features is more effective and accurate, presenting 106 fewer errors and reducing the computation time for classification in 81%, in comparision to experiments that uses a kNN on an arbitrary 44-feature set. The data used for training step for developing this method was generated and collected on a typical university network over a 24 hour period, and the testing data was captured from the same network over a 24 hour period three days later. From these raw captures, encrypted traffic generated only by Dropbox [Dro18] traffic and was extracted by filtering only traffic originating from or terminating on port 443.

- Research such as Dehghani *et al.* [DMKK10] discuss that recent applications are using unpredictable port numbers, which turns port-based traffic classification methods not effective. Thus, they proposed an approach based on Naive Bayes algorithm that uses payload content and statistical flow information for real-time network traffic classification. The authors argue that this method is suitable for traffic classification in real-time networks due the usage of only a little part of flow statistics that is enough for traffic classification. They used the MAWI dataset to validate the method and results were present on classifying HTTP, SMTP, FTP, SSH, POP3 and DNS packets with more than 80% of accuracy.

- Zhang *et al.* [ZKF+17] discuss that the increasing usage of encryption protocols in the Internet, such as Secure Sockets Layer and Transport Layer Security encryption protocols, traditional internet traffic classification approaches based on port, IP and packet content are not effective to identify the traffic flows. They proposed an approach to classify encrypted network traffic based on Metric Learning algorithms to learn the adaptive distance metric for the multiple features that can be extracted from the traffic. An artificial dataset was constructed for the training process and for validating the method based on tests that were performed. They also present results of several state-of-the-art algorithms, such as Naive Bayesian classifier, Logistic Regression, Decision Tree and Random Forest to show that their method successfully learned the adaptive distance metric of the features, and, therefore, it can take better use of the characteristics of the features for classification.

- Auld *et al.* [AMG07] present a traffic classifier that is high accurate on application types without any source or destination host-address or port information. This method uses a Bayesian trained neural network. Training data with categories derived from packet content and features were derived from packet streams that have one or more packet headers to perform the training process, as well as the validation through result analysis. Thus, this method can provide classification without access the packets content, being able to classify traffic by using only commonly available information. Traffic was classified as BULK, DATABASE, INTERACTIVE, MAIL, SERVICES, WWW, P2P, ATTACK, GAMES and MULTIMEDIA in a dataset that was generated by the authors both for the training and testing process.

4.7 Encrypted traffic analysis

As we discussed in previous sections, networks security becomes ever more important nowadays. As the usage of encryption is increasing, it is important to adopt methods that enable traffic inspection and analysis even when the data are encrypted. This section describes some methods that are being developed in this field.

- Ehsan *et al.* [MFH18] proposed a semi-supervised method based on graph theory for classifying both encrypted and plain network traffics using statistical features. This approach is based on means of concepts of graph theory, minimum spanning trees and C4.5 decision trees for classifying traffic like SSH. This method utilizes clustering algorithms and label propagation techniques. Initially, network flows are clustered and labeled based on statistical features using graph theory and minimum spanning tree algorithm. Next, some pivot data instances are selected for the expert to vote for their classes, and the identified class labels will be used for similar data instances with no labels. Finally, the decision tree algorithm is used to construct the classification model. The experimental results show that the method was accurate on classifying different network flows with suitable performance. Different datasets were used for evaluating the method, with are NLANR(AMP) [MFH18], MAWI [Groa], DARPA99 [MMIoT] and Moore [MFH18].

- A survey of methods for encrypted traffic classification and analysis is presented by Velan *et al.* [VD15]. The authors present existing approaches for classification and analysis of encrypted traffic, describing the main encryption protocols. Payload-based methods, which use knowledge of a packets' structure, and feature-based methods, which use characteristics specific to the protocol flow, are described. They also show that data extracted from the initiation of an encrypted connection and the protocol structure can be used for traffic classification and analysis. Some discussion about advantages of classification methods described in the survey is also made, such as the method's ability to identify the encrypted application protocol and the encryption protocol. A comparison of the surveyed feature-based classification methods, as well as they weaknesses and strengths, is also presented.

- Cisco [Cisc] is a big networks devices and technology vendor. Recently, the company publicized a white paper about encrypted traffic analysis that describes their strategy to manage encrypted traffic and detect malware in such traffic. Some products, such as Stealthwatch [Cisc] have been put on the market. Cisco's encrypted traffic management and encrypted malware detection approach is based on the extraction of metadata from packets and flows that are used in different classification methods that use, for example, new encrypted traffic data elements in enhanced NetFlow by applying machine learning and statistical modeling. Machine learning algorithms are used to pinpoint malicious patterns in the encrypted traffic to detect threats and improve incident response. Different data are used for such analysis, such as Sequence of Packet Lengths and Times, byte distribution, Initial Data Packet, TLS records, TLS record lengths, TLS record times, TLS record types, TLS handshake types, TLS cipher suites, TLS extensions, TLS extension lengths, TLS extension types, TLS extension version, TLS key length, TLS key session ID and TLS random. As a part of their strategy, those data are also used to define a baseline, i.e., to model a "normal" behavior for hosts and users. Then, the encrypted threats detection methods are based on correlating traffic with global threat behaviors, identifying infected hosts, command and control communication and suspicious traffic. It is important to note that these methods are intended to cisco devices. Thus, the company also developed Cisco's unique Application-Specific Integrated Circuit (ASIC) architecture, which is used to extract these data elements without overloading the network and not affecting the network's performance [Cisa][Cisb][Cisc].

4.8 Discussion

This chapter presented several works that are related to our approach. First, different approach for detecting port scan attacks are described. The methods found are useful for detecting port scan, however, they can not stop or prevent such attacks and also do not consider encrypted flows. After, generic Dos detection methods are described. Although those approaches are not intended to SDN and also not to encrypted attacks, these strategies are useful to our research, as they could be adapted and used as a partial detection step. Then, works that are identifying generic malware in encrypted traffic are presented. Next, methods for detecting different attacks in SDN are discussed, which are relevant to better understand how other research are addressing protection in SDN, what limitations they have and also what datasets they are using (usually own generated data).

Table 4.1 summarizes similar research that is intended to detect attacks. In the first column, the research is identified. Then, the contribution, i.e., the research goal is summarized in column "CONTRIBUTION", being DoS, Port scan, worms, insider detection and/or prevention methods, prediction of hosts that may be attacked in the future and also an overview of current security solutions. The column "E" identifies if the research mention encrypted attacks. The column "DATA" shows that typically own generate datasets are used in SDN research. Follow, the main technique used in the research is presented and the environment that it is intended to is show in column "ENV.". Most of the research is intended to SDN, as this thesis is also focusing in such networks. However, some approaches are intended to traditional networks (TN) and webservices, due to they goal being similar to our research. In the last column (Y) the year of publication is presented.

Finally, several traffic classification methods are discussed. Such approaches are important in network management to provide more visibility about the network traffic, which is helpful to design protection methods that can identify malicious data and anomalies. The last subsection presents some works that are intended to encrypted traffic analysis, which is a topic that is very close related to our research, as the proposed methods can be extended for detecting encrypted attacks.

Table 4.2 summarizes research that discuss and present traffic classification methods. In the first column, the research is identified. Its contribution is summarized in column "CONTRIBUTION", being presented methods that aim to classify traffic in encrypted and non-encrypted only, methods that can identify specific data on encrypted packets, such as SSH, P2P, SMTP, DNS, FTP and orthers. Other works are presenting methods that identify malicious traffic over encrypted data, and there were also found research that intend to map the most useful features for encrypted traffic classification. The column "ET" identifies if the research is intended: a)N: to classify network traffic on non-encrypted data; b)C: to classify encrypted traffic and c) CA: to classify and analyze encrypted traffic, identifying anomalies, protocols and attack patterns. The column "DATA" shows that typically own generate datasets are used in such research. Follow, the main used techniques in the research are presented and the environment that it is intended to is show in column "ENV.". In the last column (Y) the year of publication is presented.

Table 4.1 – Related works comparison

RESEAR.	CONTRIBUTION	E	DATA	TECHNIQUE	ENV.	Y
[AHR+10]	Port Scan IDS	N	Own	Algorithm based on scores	SDN	2010
[LF17]	Port scan IDS	N	Own/ Darknet	Graph analytics	TN	2017
[AM17]	Encrypted malware	Y	Own	Approach based on ML	TN	2017
[APM17]	TLS malware	Y	Own	Analysis of TLS malware	TN	2018
[ZGYW16]	Port scan IDS	N	Own	Port hopping	SDN	2016
[BNR+17]	DoS IPS	N	Own	Entropy	SDN	2017
[DDZX16]	DDoS IDS	N	DARPA	Seq. Probability Ratio Test	SDN	2016
[JGSG17]	DoS IDS	N	Real/ Traffic	CUSUM	Web/ Serves	2017
[OCD16]	DoS IDS	N	CAIDA	change-point using inter-arrival time	Cloud	2016
[NZD+16]	Predict hosts that may be attacked	N	LongTail	Machine learning	SDN	2016
[SSR15]	DDoS IDS	N	Own	OpenFlow features to correlate previous incidents	SDN	2015
[Pal15]	Overview of current solutions	N	-	Network policies management	Cloud/ SDN	2015
[GAA+14]	DoS,port scan, worms	N	Own/ CAIDA	Entropy	SDN	2016

Although some of the works in Table 4.1 are intended to SDN, they do not consider encrypted attacks. In this thesis, those works were analyzed to better understand attack detection methods and to adapt them or even develop new ones that can be used for detecting encrypted attacks in SDN, as encryption is being used increasingly. Moreover, most of the traffic classification research presented in Table 4.2 is intended to traditional networks, and some of them do not consider encrypted data. However, those approaches can be adapted and are useful to define encrypted traffic classification methods for SDN. Thus, all the works presented in this Section are important and related to this Thesis, although the most closely related ones are those described in Section 4.7.

Table 4.2 – Traffic classification methods

RESEAR.	CONTRIBUTION	ET	DATA	TECHNIQUE	ENV.	Y
[dSMB+15]	OpenFlow features selection to classify DoS,FTP,video	N	own	PCA,DFT	SDN	2015
[dSJAF+15]	Anomaly traffic classification (DDoS, worms and others)	N	own	Machine learning	SDN TN	2015
[dSJAFG16]	ATLANTIC: Anomaly traffic detection, classification and mitigation	N	own	information theory, machine learning, entropy.	SDN	2016
[LCCY17]	New method for SSH and P2P traffic classification.	C	own	Algorithm based on first N packets and C.45	TN	2017
[MSSK18]	Features selection for encrypted traffic classification	C	own	k-nearest neighbor and Fast Orthogonal Search	TN	2018
[DMKK10]	HTTP, SMTP,FTP, SSH,POP3 and DNS identification on encrypted traffic	C	MAWI	Naive bayes on payload content and flow statistics	TN	2010
[ZKF+17]	Classification method	C	own	Metric learning algorithm	TN	2017
[AMG07]	Method to classify Bulk,database,P2P, www,interactive,mail, services,attacks, games and multimedia traffic on encrypted packets	C	own	Bayesian neural networks	TN	2007
[MFH18]	classify SSH using statistical features	CA	NLANR, MAWI, Moore, DARPA	Graph theory, C.45, minimum spanning tree	TN	2018
[VD15]	Survey of available methods	CA	-	-		2015
[Cisc]	Encrypted malware detection	CA	own	Machine learning and statistics	Cisco /TN	2018

The usage of cryptography is growing specially in enterprise network traffic, creating new challenges to security leaders on protecting networks and systems against enemies that can hide malicious activities in encrypted traffic. Gartner [1] discuss that in 2019 80 % of the Internet traffic will be encrypted and half of campaigns will use cryptography to conceal malware delivery, data exfiltration and command and control activity [Cisc]. Thus, encrypted traffic analysis is a core topic that must be investigated, as it is a key step on identifying anomalies in such traffic, to provide more security visibility by performing real-time analysis correlated with user and device information to analyze threats behavior in encrypted traffic and contextual threat intelligence. Big network devices and systems vendors, such as Cisco [2] are currently working on devices and systems that can analyze encrypted traffic for detecting hidden malware in such data [Cisc, AM17, Cisa, Cisb].

[1]Gartner (gartner.com) is a global research and advisory company that provides insights, advice and tools in IT security and others.

[2]Cisco (www.cisco.com) is a big network device and systems vendor.

5. DETECTING ENCRYPTED ATTACKS IN SDN

Several attacks can affect SDN architectures (see Section 2.3). Thus, this chapter presents our protection methods for detecting encrypted DoS, port scan and generic attacks. Several SDN components could be compromised. For example, the SDN controller and its interactions with other SDN components (*e.g.* switch) could be affected by an adversary through different attacks. Therefore, we first describe a generic attack modeling method to better define and understand what attacks can be performed in SDN.

5.1 Attack model

In order to implement a protection system, it is important to define an attack model to better understand how a target could be compromised. Thus, in this thesis we are using the STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege) model [SWJ15], which is also applied in other research related to SDN security [Foud] [DKMB15] and is suitable for our approach. STRIDE is also adopted in traditional network security research [XTS+12] [CCAC11] [Tü17].

The STRIDE model considers the following attacks:

- Spoofing (S): when an attacker impersonates another legitimate user or program.
- Tampering (T): an attacker modifies application resources, such as in memory data.
- Repudiation (R): when users, legitimate or not, denies their actions.
- Information disclosure (I): the information/data are exposed and available to unauthorized person or system.
- Denial of service (D): an attacker overloads a system resource or a network in order to turn it unavailable.
- Elevation of privilege (E): a malicious user get privileged access to resources that should be protected and restricted.

Each type of attack may affect specific parts of a network and/or system, *i.e.*, may have different goals that can, for example, impact some basic security components like authentication, integrity, confidentiality, availability and non-repudiation, as discussed in Section 2.3. Typically, those attacks are intended to specific targets in SDN, such as the controller, a network element (NE) or an application that is running on the controller. Table 5.1 shows how different attack types can affect SDN, according to the STRIDE model [SWJ15]

[Foue]. The affected properties (authentication, integrity, non-repudiation, confidentiality, availability and authorization) are described in Chapter 2.

Table 5.1 – STRIDE attack model applied to SDN

ATTACK TYPE	**AFFECTS**	**SDN TARGET**
Spoofing	Authentication	NE, Controller, Applications
Tampering	Integrity	NE, Controller, Applications
Repudiation	Non-repudiation	Controller
Information disclosure	Confidentiality	NE, Controller, Applications
Denial of service (DoS)	Availability	NE, Controller, Applications
Elevation of privileges	Authorization	NE, Controller

Although several attack types can affect SDN, in this research we focus on information disclosure, which affects the system confidentiality, and DoS attacks, which affect the system availability. Both attack types can be launched against an SDN network element, a controller or even applications.

Information disclosure attacks against SDN typically precede another attack. This can be made by different methods in SDN, depending on the target:

- **Information disclosure on network elements**. To perform information disclosure on SDN network elements, *e.g.* a switch, an attacker could discover the switch storage capacity by filling its flow table. Moreover, a port scan attack could also be used to discover what ports the switch has and some information about its configuration and vendor. Finally, statistical information could be collected by an attacker on the switch to discover its capacity and traffic behavior. With this information, other types of attacks could be lunched, such as a DoS, by flooding a low-capacity switch with a large number of new bogus flows or send a large amount of data to ports that have weak forwarding capability.

- **Information disclosure on a controller**. An adversary can get some important information about an SDN controller, such as opened ports, services and applications that are running and operation system versions by performing, for example, a port scan attack. Typically, such attack is used to retrieve this information that allows to lunch another attack, such as a DoS. Another risk of information disclosure in SDN related to the exposure of cryptography keys used for privacy or authentication. Usually, the controller holds keys that can verify switches identity. In SDN, those keys should not be transferred through the network via messages. *Braadland* [Bra17] proposes a Key Management scheme for Data Plane Encryption in SDN as an extension to the southbound communication protocol to address such vulnerability.

- **Information disclosure on applications**. To perform information disclosure on SDN applications, an attacker could observe the message exchange between the controller and applications to get important information, such as the application IP address. Thus, to protect SDN applications from information disclosure, the communication between the application and the controller should use encrypted messages.

behavior analysis

The main DoS attacks that can affect SDN networks are based on attacking a network element (*e.g.* switch and servers), the controller or even applications running on the controller. This can be done in several ways, also by using encryption to empower the attack:

- **DoS on network elements**. In SDN, connected devices can be attacked on a similar way that in traditional networks, *i.e.*, by overloading hardware resources. Therefore, protection systems should be used to monitor core devices, like servers. Another core network element in SDN is the switch, which is a hardware that usually has limited resources (*e.g.* storage capacity). Therefore, it can only store a limited number of records in its flow table and an attacker could overload the switch's memory on a way that the flow table becomes full. Thus, the switch would not be able to add new entries and would reply error messages to the controller as well as to drop legitimate packets. The controller has to handle every useless new flow for flow entry creation, which increases hardware resource consumption and can decrease the controller responsiveness to legitimate flow requests. For example, an attacker could perform a DoS attack by changing the flow table entries on a way that multiple data flows from different ingress ports are forwarded out from the same egress port, exceeding its capabilities. Another way to perform an encrypted DoS on an SDN switch, is to send a large number of bogus messages/packets in a short time frame through the encrypted channel between switch and controller, as the processing of such messages increases the processing power consumption due to encryption and decryption processes.

- **DoS on the controller**. Attackers can compromise an SDN switch or application in order to gather information about the controller, such as open ports and running applications. This could be done, for example, by performing a port scan attack. With this information, attackers could overload the controller by exchanging a large number of concurrent messages with the controller on the discovered ports and applications, also performing a DoS attack. Usually, in centralized SDN networks, one controller manages a large number of switches and applications and an attacker may be able to overload this controller by sending massive useless packets. This process can be empowered when encrypted data are sent through the encrypted channel between the switch and controller, as the encryption and decryption processes increase the processing power consumption. Whenever a packet arrives on a switch and a table miss occurs (no flow entry that matches it is found) it is forwarded to the controller, which starts a new flow creation process. Thus, an attacker could also be able to compromise SDN switches to forward a large number of missed packets (flow table miss), overloading the controller and performing a DoS attack. Another way to perform a DoS against an SDN controller is to compromise running applications that continually request hardware resources to the controller, such as CPU and memory, overloading these resources.

- **DoS on applications**. Usually, applications that are installed in the control plane and provide some service to SDN users and devices could also be overloaded by an attacker on a way that those applications became unable to process legitimate requests as required. Similar methods that are found for traditional networks [OCD16] [JGSG17] [YY15] could be used to overload SDN applications.

5.2 Encrypted attacks detection possibilities in SDN

Usually, an SDN signature-based IDS (see Section 2.4) typically cannot analyze encrypted packets, because they need to analyze the payload data that is encrypted. However, anomaly-based IDS (see Section 2.4) may be applied, using three main approaches: protocol-based, modification-based or statistical-based [KGR14], that could be used not only in SDN, but also in traditional networks. Figure 5.1 shows a possible taxonomy for methods that can be used for detecting encrypted attacks. These methods are grouped according to the basic approach:

- **Protocol-based analysis**. This approach, also know as stateful protocol analysis, searches deviations from the packets in each state of the protocol. Universal profiles that specify how a given protocol may or may not be used in data transfer are analyzed. They are based on the protocol specification of software and hardware vendors, and also official protocols standard. However, since this type of approach only analyses whether the protocol is being applied in a proper way, it is not possible to detect attacks that are being performed at the application layer, which are the most widely used. Therefore, a combined approach that first identifies the protocol, such as the application protocol, and then analyzes whether it has a normal behavior or not. This analyzes could take into consideration the protocol specification, for example, packets length, inter-arrival time, bytes per packet and packets per second.

- **Modification-based analysis**. This approach to detect encrypted attacks consists on two basic ideas:

  – Changing the encryption protocol and infrastructure to detect attacks in encrypted data on the network. Basically, the key (password) to cipher and to decipher the

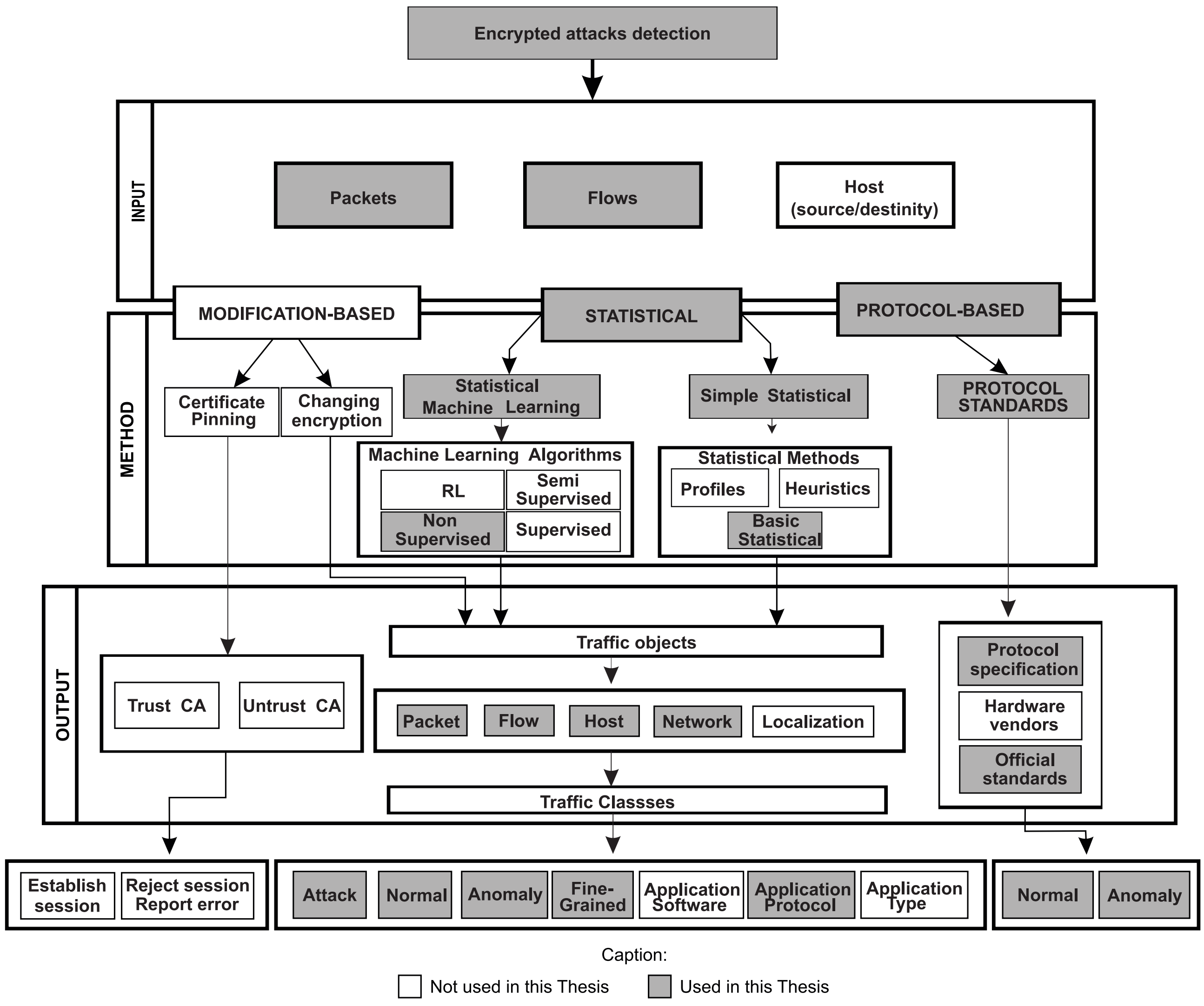

Figure 5.1 – Taxonomy for encrypted attacks detection

data are distributed to the IDS. With this secret, the IDS can decipher the package payload and analyze it. However, this technique can turn the network vulnerable and the privacy principle may be broken. Therefore, such methods should only be used for private networks, such as a private communication between different branches of a company. In addition, it may consume a lot of processing power, which may turn this method slow and even to make it impossible to apply in real environments with large data traffic.

– Certificate pinning is another strategy that can be used for encrypted traffic management, to prevent malicious encrypted attempts. Usually, in the handshake process, to establish an SSL/TLS connection a client can authenticate the server by validating the server certificate. This validation process is based on the confirmation that the server certificate was issued by a known and trusted Certificate Authority (CA). Thus, client devices or applications are previously configured to recognize a trusted server certificate and drop flows/packets if a trusted CA cannot be identified.

- **Statistical-based analysis**. It is also possible to develop encrypted attack detection methods using statistical analysis. Such analysis is based on:
  - Observable parameters (metadata) of encrypted data traffic, such as flow information (*e.g.* the number of transmitted or received packets per second, bytes per second, inter-arrival time and packets with errors).
  - Network Behavior Analysis (NBA) [Koc09] methods to identify anomalies in relation to a normal behavior established previously, using, for example, machine learning algorithms. Some information, like source and destination IP address and ports, the header fields and payload size can be used as input for such methods.

  Statistical-based analysis for detecting encrypted attacks can be performed by using methods based on statistical machine learning to identify anomalies in the network traffic. Machine learning algorithms can be non-supervised, supervised, semi-supervised and Reinforcement Learning (RL) [VD15]. Statistical methods usually are based on traffic profiles, heuristics and basic statistical data. Those methods can be used to retrieve information about packets, flows, communicating hosts, networks and also the location of those hosts (*e.g.* country) to define rules for traffic management. Then, detection methods can be used to:

  - detect specific attack patterns;
  - establish a normal traffic behavior or resource usage;
  - detect anomalies in the traffic behavior or resource usage;
  - traffic classification, which can be used to identify the packet protocol (*e.g.* transport layer protocol, network protocol and even the application protocol). Fine-grained methods can also be developed to identify more details about the traffic, such as the application software or type.

5.3 Our approaches for detecting encrypted attacks in SDN

This section describes a lightweight anomaly-based IPS that uses the switch counters to detect and prevent port scan and DoS attacks to an SDN infrastructure, even if the flows/packets are encrypted. Although several attacks could compromise an SDN network, in this work we focus on DoS and port scan, since they are the most common attacks. We also present an approach for detecting generic attacks in encrypted traffic that could be applied against an application in SDN.

On SDN, the controller has a global view of the network and may block malicious flows closer to their source. In the developed proposal, the controller manages and stores the rules that define packet forwarding on the network in addition to collecting counters data from SDN switches, as can be seen in Figure 5.2.

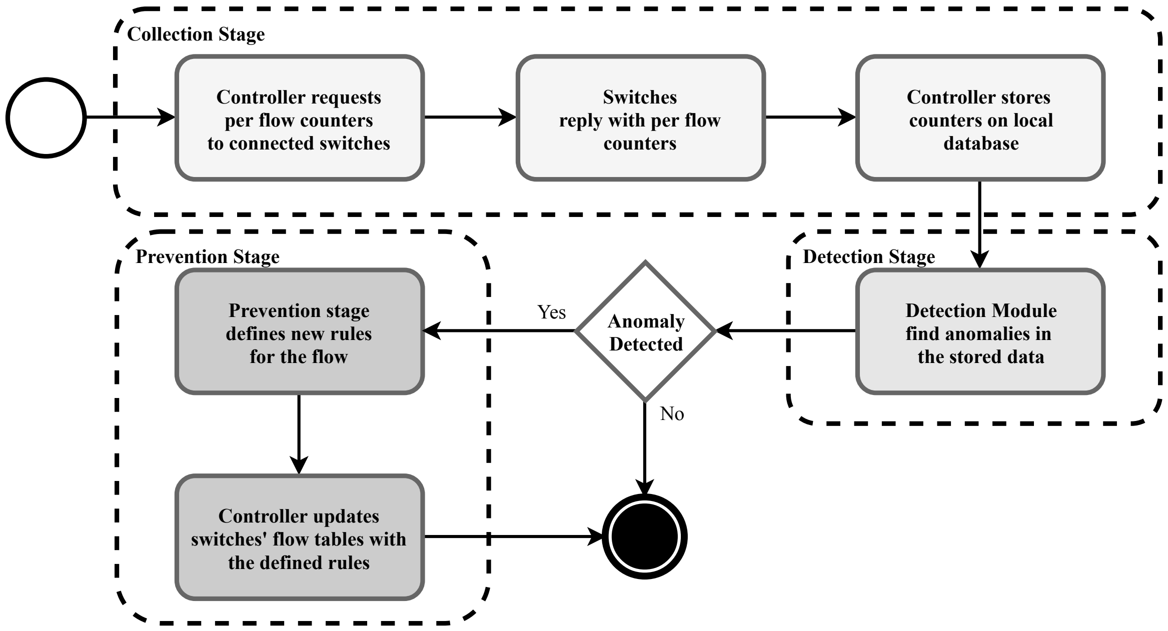
Collection Stage
Controller requests per flow counters to connected switches
Switches reply with per flow counters
Controller stores counters on local database
Prevention Stage
Prevention stage defines new rules for the flow
Controller updates switches' flow tables with the defined rules
Yes
Anomaly Detected
No
Detection Stage
Detection Module find anomalies in the stored data

Figure 5.2 – IPS operation flow

Our solution was developed according to three basic stages. The first one is the Collection stage, which performs data collection on the switches flow tables. Basically, the controller requests per flow counters to the connected switches; waits for the switches to reply with the per flow counters; and finally, the controller stores the counters in a local database. In the second stage, detection is performed on the data collected on the previous stage. Finally, in the third stage, the prevention method is implemented to react to attacks based on the Detection stage results. In the last stage, if an anomaly is detected, then new rules are defined and included in the switches flow tables. Then, the switches are responsible for dropping new similar malicious flows based on those rules. Each of these stages will be further described in the next sections.

5.3.1 Collection stage

As our protection methods are based on switches counters data, in the first stage, a job periodically requests flow counters to the switches and stores them on a database. This job, which is a script, and the database are located on the controller. The statistic collection interval is a critical point to be considered. On the one hand, if the collection is performed between wide time intervals, there will be a delay in the attacks detection. On the other hand, if the time interval is too narrow, there will be an increase on traffic related to collection requests. In this work, this interval was defined as three seconds, as it was considered appropriate based on our test results, as this interval did not cause any network overload. This three seconds interval is also the default value set by OpenDayLight (used in the next Chapter on our experimental evaluation). This threshold can be set according to the network and detection requirements in the get_counters() function (Method_A) and Time$_I$*ntervalvalueinMethod_BandMethod_C*.

Through an API, the controller sends a message to the switch asking for flow counters and the switch replies this message with one or more messages that contain all the counters data. On the controller, an application reads these data and extracts the information that will be used by the IPS. This information contains the source and destination addresses, ports and the number of packets from each flow. The counters are updated in the flow table every time a new packet is received. Those counters are used for generating statistics, in order to monitor the number of packets and bytes of each stream, in addition to the duration time since its inception.

5.3.2 Detection stage

In this stage, specific methods must be applied to perform detection against attacks. According to our scope, we only present methods intended to port scan and DoS in this section. However, other methods could be used in this stage based on the data collected in the previous stage. A blacklist is also implemented on the controller, which is used to store the source of flows that were identified as malicious.

Table 5.2 summarizes the methods being proposed in this thesis. The "DETECTION" column shows the detection method to identify specific attacks showed in column "ATTACK" when performed against a target, as described in column "TARGET". The last column, shows what strategy is used to implement detection.

Table 5.2 – Methods being proposed in this thesis

DETECTION	**ATTACK**	**TARGET**	**BASED ON**
METHOD_A	Port scan	Controller, switch	Statistics
METHOD_B	DoS	Controller, switch	Statistics, CUSUM
METHOD_C	Generic attacks, insider	applications, hosts	Statistics, ML, protocol classification, behavior analysis

A) Detecting port scan attacks

To perform port scan detection, the flow information stored in the database on the previous stage is analyzed. We called it METHOD_A and each analysis consists on the following steps [NTL+18]:

- Selection of stored flows that have a reduced number of transmitted packets (which may characterize port scanning flows) and that have been generated in the last three seconds;
- Grouping of the selected flows, by source and destination IP and destination port;
- Obtaining the number of flows with similar source and destination addresses in a small interval that must be defined as required (for example, 1s). Eventual port scan attempts may occur on any system, so a single stream cannot be considered an attack;
- Scan categorization (horizontal, vertical or mixed) based on source and destination addresses, destination port and packet count;
- If a port scan is detected, the source IP is added to a blacklist. All the flows from an address stored in this list will be dropped.

Only one port scan is not considered an attack attempt, so that, in order for a stream to be considered an attack attempt, some rules have to be met, depending on the type of performed port scan. For example, source IP addresses that have streams with number of packets lower than a given threshold, and that have at least a number of target hosts with same destination port will be considered an anomaly that represents a horizontal port scanning. Thus, the source address is added to the blacklist.

We implemented this detection method as follow. First, a function *getCounters()* collects the switch's counters and store it in a database. Then, the required thresholds must be set. After, a function called *computeScansPerHost()* performs horizontal port scan detection. Finally, the result of this function, stored in Scans_per_Host, is compared to the defined threshold to perform detection. If this value is above the threshold, an horizontal port scan attack is detected. Thus, *writeBlacklist()* function is used to write an entry in the blacklist, *defineDropRules()* defines the flow must be discarded and an alert is send to networks administrator and also logs are written by the function *logAndAllertHorizontalScan()*. No further analysis is made and the algorithm stops. However, if no horizontal scan is detected, it is still necessary to check vertical scans. Therefore, function *computeTargetHostScanned()* is used to compute the number of scans on common ports and other ports, according our weight-based method. If the total weigh computed (Target_Host_Scanned) is above the defined threshold (15) an horizontal port scan attack is detected and same functions as used when a horizontal port scan is detected are used to provide protection. The entire port scan detection method is show in Algorithm 5.1.

Algorithm 5.1 Detecting horizontal port scan in SDN

```
Require: Target_Host_Threshold, Scans_per_Host_Threshold;
  getCounters();
  computeScansPerHost();
  if Scans_per_Host > Scans_per_Host_Threshold then
    writeBlacklist();
    defineDropRules();
    logAndAllertHorizontalScan();
  else
    computeTargetHostScanned();
    if Target_Host_Scanned > Target_Host_Threshold then
      writeBlacklist();
      defineDropRules();
      logAndAllertHorizontalScan();
    end if
  end if
```

The number of target hosts needed to characterize the attack attempt can be adjusted according to the network requirements on which our solution will be used. In the vertical port scanning detection, it is assumed that a source host performs the scanning of several ports on the same destination host. In this case, source hosts that have streams for different ports on the same target hosts may be recognized by this application as an anomaly.

In order to detect a port scan, a weight-based method was created to allow greater sensitivity to the attack attempts for the most used ports. Typically, the most common probed ports, such as Telnet, FTP or SMTP, have a higher weight compared to other ports. The ports that should have a higher probability to being scanned are 22, 23, 25 and 3389, as they are the most scanned ports according to the statistics of incidents reported by CERT.br [CER18]. However, this group of common ports must be defined, as required by the network.

Our detection method computes the sum of the total weight from flows of the same source and destination. If this sum exceeds a threshold, the flows are considered as port scan, then the source IP is added to the blacklist. This threshold must be properly set according to the network, as a higher threshold may reduce the method effectiveness, since an attacker may be able to perform more scans before being detected, and a lower threshold, could compromise legitimate access and present a high number of false positive detection. Whenever the sum of weights from flows of the same source and destination exceeds the threshold, an attack may have been detected, and the source address is added to the blacklist. Equation 5.1 is used for computing the amount of connections or attempts from a given source to be considered as a port scan. The calculated value TotalWeight is compared to the defined threshold for detecting a port scan.

$$\textbf{TotalWeight} = \textbf{CSP\_Weight} * \textbf{CSP} + \textbf{OSP\_Weight} * \textbf{OSP} \tag{5.1}$$

Where:

- TotalWeight is the value being calculated to compare to a defined threshold for detecting a port scan attack
- CSP indicates the number of scans on common ports
- OSP the number of scans on other ports
- CSP_Weight is the weight of common scanned ports
- OSP_Weight is the weight of other scanned ports

For mixed port scans, both assumptions already mentioned must be considered. Our IPS analyses the source hosts that attempt to establish connections on at least the value set for the threshold Target_Host_Threshold hosts and at least one of these hosts must also present a sum of vertical port scan weights higher than the threshold set for TotalWeight in Equation 5.1. In a nutshell, if Target_Host_Threshold hosts have ports scanned and at least one of these has a different port scanned, it is considered a mixed port scan [NTL+18].

B) Detecting denial of service attacks

A common attack that an adversary can perform in SDN is a DoS. Thus, our approach is intended to detect such attacks against the switch and controller when an attacker sends a large number of bogus OpenFlow packets in a short time frame through an encrypted channel. Usually, when an SDN switch or a controller is under such attack, there will be a burst increase in packets sent and also in processing power consumption due to the encryption and decryption process. Silva *et al.* [dSMB+15] present a study that identifies the most appropriate flow features that can be used to classify DoS attacks, FTP traffic and video streaming.

Although several features can accurately represent DoS flows, we chose the number of packets per second and the number of bytes per second. We chose those features to use as input in our detection method due to the characteristics and particularities of how such attacks can be performed, and based on experimental evaluations (see Section 6.3). Each feature is computed individually, and an attack is detected if at least one computed feature observed on the analyzed samples has an abnormal behavior. Those features can be requested on the switches flow table (counters) in SDN, and could also be computed or even collected on traditional networks MIBs (Management Information Base) [XC18] through SNMP, for example. To detect such attacks, we developed a method that we called as

METHOD_B. This method is based on the CUSUM algorithm [MRFFFRAB15], which is typically used as a sequential method to detect anomaly traffic behaviors [JGSG17] [OCD16]. It is based on adding the difference between the current sample value and the collected samples average, as presented in Equation 2, where:

- C_i: is the current value of the cumulative sums
- C_i-1: the previous value of the cumulative sums
- X_i: is the value of the current sample being computed in the CUSUM
- Average_X: is the average value of the samples that have already been computed
- N: is the number of samples
- i: is a counter for the number of samples

$$C\_i = C\_i - 1 + (X\_i - Average\_X) \quad \text{for} \quad i = 1, 2, 3, \dots N \tag{2}$$

The CUSUM algorithm concept is that its computed value over a samples sequence during the normal condition is negative and only becomes positive when a significant change occurs [MRFFFRAB15] [CB15] [OCD16]. Hence, we customized this assumption on a way that the monitored features can be used for detecting encrypted DoS on a switch or a controller in SDN. Thus, we consider that the CUSUM value during a normal traffic behavior must be lower than a defined threshold. When the CUSUM value is above this threshold, the sample is considered abnormal. However, the method only detects a DoS attack when it exceeds a predefined anomaly samples threshold. Based on this behavior, a high false positive rate is avoided, as only one anomaly sample can be an outlier, which is caused by the network and does not implies that an attack is happening. First, we use the selected features (bytes per second and packets per second) fetched from the switch in the Collection stage, individually, as input in our detection algorithm. The following thresholds must also be defined, according to the network requirements:

- Thresholds used to compute bytes per second:
  - C_Threshold_BS: the maximal difference percentage on bytes per second between a sample and the average of samples
  - Max_Anomaly_Samples_BS: the bytes per second samples percentage that are above the defined threshold C_Threshold_BS
  - Time_Interval: the time interval to analyze bytes per second samples (in seconds)
- Thresholds used to compute packets per second:

  - C_Threshold_PS: the maximal difference percent on packets per second between a sample and the average of samples
  - Max_Anomaly_Samples_PS: the packets per second samples percent that are above the defined threshold C_Threshold_PS
  - Time_Interval: the time interval to analyze packets per second samples (in seconds)

METHOD_B performs detection based on those thresholds and stored samples. Each time an abnormal sample is found, that is, the computed CUSUM value is above the threshold, a counter is incremented and the sample value discarded from the CUSUM computation so that the following samples can be correctly analyzed, because only one anomaly sample cannot be considered an attack (it could be some noise or failure, for example). However, if the computed anomaly samples number in a time frame reaches the threshold, an attack is detected.

In order to compare the results from the CUSUM algorithm, we also have developed a method that only computes the difference between the sample being analyzed and the average of samples that have already been computed and considered normal traffic behavior samples. This method produced results that are similar from the CUSUM algorithm. In our experiments (see Chapter 6), we have observed that traffic behavior, *i.e.*, normal and anomalous, can be characterized by this difference in relation to a defined threshold. Thus, in this method we also use the selected features (bytes per second and packets per second) fetched from the switch in the Collection stage, individually, as input in this detection algorithm. Hence, METHOD_B performs detection based on those thresholds and stored samples. Each time an anomaly sample is found, that is, the computed difference is above the threshold, a counter is incremented. However, if the difference between the sample that is being analyzed and the average of samples that have already been analyzed is lower than the threshold, this is considered a normal behavior and the anomalous samples counter is not incremented. If the computed anomaly samples in a time frame reaches the threshold, an attack is detected. This anomalous samples threshold must be set properly according to the network, because only one anomaly sample (or only a few) cannot be considered as an attack (it could be some noise or failure, for example). The number of samples that are analyzed is defined by the values set for the time interval to analyze packet-per-second and byte-per-second samples, and the collection interval is defined in the Collection stage. Follow thresholds must also be defined, according to the network requirements:

- Thresholds used to compute bytes per second:
  - D_Threshold_BS: the maximal difference on bytes per second between a sample and the average of samples in relation to the samples average (in percent)
  - Max_Anomaly_Samples_BS: a threshold for byte-per-second samples that are above the defined threshold D_Threshold_BS (in percent)

  - Time_Interval: the time interval to analyze byte-per-second samples (in seconds).

- Thresholds used to compute packets per second:
  - D_Threshold_PS: a threshold value for the maximal difference on packets per second between a sample and the average of samples (in percent)
  - Max_Anomaly_Samples_PS: the packet-per-second samples that are above the defined threshold D_Threshold_PS (in percent)
  - Time_Interval: the time interval to analyze packet-per-second samples (in seconds)

Thus, our DoS protection method is composed by four main steps, being:

- Analyze packet-per-second by our customized CUSUM
- Analyze bytes-per-second by our customized CUSUM
- Analyze packet-per-second by computing difference between a sample and the average of samples
- Analyze bytes-per-second by computing difference between a sample and the average of samples

In each step, detection is performed by routines computePSCUSUM() and computePSDiff(), which compute the cumulative sums strategy and difference over packet-per-second, respectively, and functions computeBSCUSUM() and computeBSDiff(), which compute the cumulative sums strategy and difference over bytes-per-second, respectively. Each routine's has an boolean output (PSCUSUM, BSCUSUM, PSDiff and BSDiff) that indicates if an attack was detected. If at least at one routine an attack is detected, the algorithm proceeds to protection, i.e., write an entry in the blacklist (writeBlacklist()), create drop rules for the flows identified as malicious (defineDropRules()), writes a log and generates an alert to the networks administrator (logAndAllert()). However, if no attack is detected, the routine defineForwardRules() indicates the controller that forwarding rules can be defined. The entire DoS detection process is shown in Algorithm 5.2.

Algorithm 5.2 Detecting encrypted DoS in SDN

```
Require: Time_Interval, C_Threshold_BS, C_Threshold_PS, D_Threshold_PS;
Require: D_Threshold_BS, Max_Anomaly_Samples_BS, Max_Anomaly_Samples_PS;
  getCounters();
  computePSCUSUM();
  computePSDiff();
  computeBSCUSUM();
  computePDDiff();
  if (PSCUSUM = true or BSCUSUM = true or PSDiff = true or BSDiff = true) then
    writteBlacklist();
    defineDropRules();
    logAndAllert();
  else
    defineForwardRules();
  end if
```

C) Generic attacks

As discussed in Section 5.1 other attacks than DoS and port scan can be performed in SDN, hence we also developed a method for detecting generic encrypted attacks. We called it METHOD_C and it is composed by following components:

- Metadata collector: this component is defined in the Collection stage to get switch counters that could also be customized and new ones could be implemented by using P4 [BDG+14] [Cis15], which allows to program the Data plane.

- Traffic classification: this component is responsible to identify the network traffic being exchanged. It can identify the application, transport or application protocol, and even application type or software (see Table 2.7), as required, by using non-supervised machine learning methods, as shown in Table 2.6.

- Normal traffic behavior establishment: a machine learning method can be used to model a standard and normal network behavior based on flow and packet information (see Section 2.5). This method can collect the required information in the network for a previously defined time (*e.g.* one day) and build the expected network standard traffic behavior.

- Set of allowed protocols: based on the normal traffic behavior, a set of protocols that are allowed in the network (*e.g.* application, transport or protocol, and even application type or software) must be defined. For this task, the traffic classification method that will be used (see Table 2.6) is a core component.

- Blacklist: this module is composed by a table with entries that can identify sources that are not allowed to communicate in the network. This component is described in the Protection stage and is the same that is also used by METHOD_A and METHOD_B.

The approach being proposed could be used for detecting several attacks in SDN, such as insider attack against an application related to data evasion or sabotage, DoS, port scan, DoS against an application and others. Although this method could also solve problems tackled by METHOD_A and METHOD_B, those are specific for DoS and port scan, and, therefore, they perform better than METHOD_C. Furthermore, since METHOD_C is a generic method, it has to be configured to different environments and contexts. Cisco [Cisc, Cisa, Cisb] is also working in a similar approach for traditional networks, using proprietary protocols, software and hardware. Figure 5.3 illustrates our encrypted attacks detection and prevention proposal workflow.

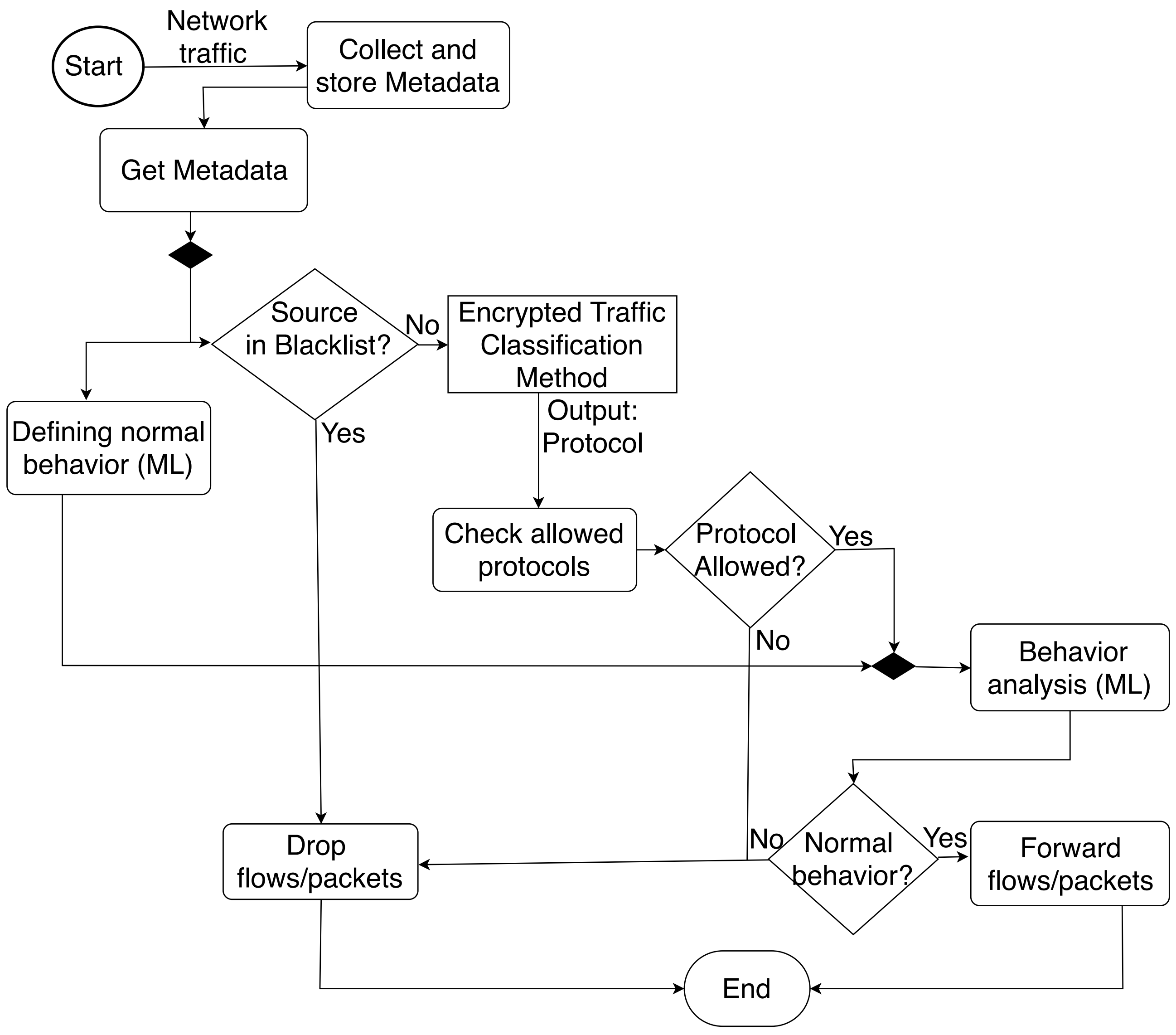

Figure 5.3 – Approach for general encrypted attacks detection

Initially, a normal traffic behavior must be established and the set of allowed protocols and/or applications must be defined. The detection process is based on the metadata stored periodically in a database by the switches counters collector process. The first detection step is to extract the source IP of the data being analyzed and match it to the entries in the blacklist. If a matching entry is found, a dropping rule is defined in the switches and the flow is then discarded. Such dropping rule is stored without a time to live stamp, so it can only be removed by the networks administrator. If no matching entry is found in the blacklist, the traffic classification process starts to identify the protocol used or even the application type or software. Then, this information is used to match with a set of allowed protocols and applications that must be defined previously. If a matching entry is found, it proceeds to the behavior analysis to compare it to the previously established normal behavior. If no anomaly is found, a forwarding rule is defined to the switches. Otherwise, a dropping rule is set and an entry is written in the blacklist. Both normal and abnormal behaviors can be classified by using machine learning algorithms [ZKF+17]. The most appropriate machine learning algorithms found in the literature [ZKF+17] are the Bayesian classifier [AMG07], the Logistic Regression [KKB07], the Random Forest [ZKF+17] and the Decision Tree [XL09]. However, those algorithms can be adapted or others can be used, as required. As input, the most appropriate features from the stored SDN metadata (counters) can be defined according to Silva *et al.* [dSMB+15], but other features can also be included to perform traffic classification. Moreover, specific counters could be defined on the data plane by using P4, as required to improve the traffic classification process and also the normal and abnormal behavior classification.

5.3.3 Prevention stage

Initially, all SDN switches have empty flow tables. Thus, upon receiving the first packet, a table miss will occur and the packet will be sent to the controller through a PacketIn message. On the controller, a PacketIn handler is used to read the packet, analyze the header and extract the source and destination IP addresses and ports to define forwarding rules. First, this handler has to check if the incoming flow has a match in the blacklist. If a matching rule is found in the blacklist, a drop rule is then defined to the switches flow table so that new incoming malicious flow sources are dropped without further analysis. If no related entry is found in the blacklist, a new one is created and added in the switch flow table to provide forwarding to the other packets related to the same stream.

Flow entries can be removed from flow tables through a request from the controller and by timeout. The timeout mechanism is executed by the switch independently from the controller and is based on the configuration and state of flow inputs. We defined a timeout for each stream to avoid overflow on flow tables. A flow will only remain on the table for

15 seconds after its last packet was received. Arriving packets on the controller usually correspond to the first packet of a new flow or, depending on its type and on the application, the controller can decide to install a rule on the switch so that all packets of a given flow are sent to the controller to being handled individually. This last case corresponds, usually, to ICMP, DNS and DHCP control packets or routing protocols like OSPF and BGP.

By defining forwarding rules, in addition to react to the attack attempt, it also protects against new flows from the same source, without sending a request to the controller. Some prevention measures were also added on the packetIn handler. For example, by default every non-malicious connection must be initialized with a SYN flag. Therefore, when receiving a packet with a flag that is different from SYN, the controller will discard the packet. This prevents other types of attacks (see Section 2.3).

When a flow is added by the controller in the flow table, its packet header information is included on the rule, so that each received stream will have a single entry in the table. A fifteen seconds timeout is also set, which allows the flow to be kept in the table if there is a small delay in packets transmission. Table 5.3 shows a flow table with the rules used by our method. Unused rules were omitted from the table.

Table 5.3 – Example of some flow table records

MATCH RULES				ACTION	TIMEOUT
ADDRESS		PORT			
SRC	DEST	SRC	DEST		IDLE
10.0.0.11	10.0.1.124	36987	80	forward	15
10.10.0.45	10.10.2.43	23234	80	forward	15
10.0.0.114	10.10.2.29	35455	22	forward	15
10.10.2.24	10.0.0.12	32444	25	drop	0

In the example in Table 5.3, the first three streams are forwarded and their idle_timeout was set to 15 seconds, so that if no related packet is received in that time, the stream will be removed from the flow table. The fourth flow has as action its discarding, *i.e.*, when a packet is received and its header fields match the rules, this packet will be discarded. The fourth flow also has the timeout set to null, which means that this is a permanent rule and can only be removed from the table by the controller through a specific message. Thus, if an attacker starts a DoS attack after a port scan is detected, even if the packets payload is encrypted, it will be blocked by our IPS, which discards all received flows from the attackers source.

The global view nature of the network enabled by SDN allows the application to scan individual switches and send discarded streams to others, making it impossible for such streams to take different routes to the target host. In addition, different controllers can exchange information about malicious flows and sources, extending protection to other networks, as a collaborative intrusion detection mechanism.

5.4 Discussion

In this chapter, the STRIDE attack model is used to discuss the main attacks that SDN is vulnerable to. As this thesis goal is to discover how encrypted attacks can be executed and identified in SDN, we show that DoS attacks could be empowered in SDN when encryption is used, by using encrypted packets that are send through the encrypted channel between switch and controller. A large number of such messages increases processing power consumption both on switch and controller due to encryption and decryption process.

Then we discussed about encrypted attacks detection possibilities in SDN. Three approaches are described: protocol-based, modification-based and statistical-based approaches. Thus, considering the inflexibility on the current network equipment, concerns on abstracting network functions of switches dedicated to applications in SDN have been increasing. Security policies can be set by the controller as rules in the flow tables [KF13] instead of manual and independent configurations. Thus, a switch provides only the filter function according to the rule in the flow table, not significantly influencing the performance of the network. Furthermore, SDN has natural statistics features that are useful for intrusion detection analysis, so that the controller gets more visibility of the network traffic. Thus, this chapter also presented our methods to identify theencrypted attacks in SDN. Those methods are statistical-based, which are the most appropriate approach to identify the attacks that are specific to SDN because natural statistics are provided in such networks that can be used to identify anomalies.

- Port scan: a weight-based method according to the port number group (common ports and other ports) is presented. When a defined threshold for scan attempts is reached, an attack is detected. This method also identifies if a host is being scanned by computing the number of scans per host. Although a port scan is not an encrypted attack, it is typically used as a first step to perform such attacks. Thus, by detecting and protecting against port scan in SDN, our method also avoids other attacks (that could be encrypted) from same source to materialize.

- DoS: a method to detect denial of service attacks against switch and controller is presented. This method can detect such attacks when a large number of bogus encrypted messages is sent through the encrypted communication channel between a switch and the controller. This kind of attack overloads the channel and increases the power consumption on both switch and controller.

- Generic encrypted attacks: we also present a generic detection method that can be used, for example, to detect insider attacks that perform data evasion or sabotage also by using encrypted data. Our approach is based on defining a set of allowed protocols (*e.g.* application protocols or applications type), establish a normal traffic behavior by using a machine learning algorithm. To perform detection, the method first uses an encrypted traffic classifier that can be based on packets or flows (when traffic classification is based on packets in SDN, P4 should be used to customize switch counters) to identify the packets/flows protocol and match with the set of allowed protocols. If a match occurs, a machine learning method is used again to analyze if the traffic has a normal behavior or not, based on the previous normal behavior established. If an anomalous behavior is detected, the flows are dropped. Otherwise, a forwarding rule is generated to allow the traffic.

Whenever an anomaly or attack is detected, an entry is written in a blacklist and drop rules are defined to the switches flow tables. Moreover, logs and alerts are generated. Although the developed methods are intended to increase protection against encrypted attacks in SDN, they can be extended to traditional networks. However, the used flow features must be calculated in traditional networks. This could result in high processing power consumption and also in delays in forwarding packets. Therefore, the hardware and network bandwidth consumption must be analyzed to define if such approach is suitable for other networks than SDN.

6. CASE STUDY

This chapter describes our test strategy to evaluate our methods, as well as the experimental network that was implemented in a laboratory environment. We also discuss the test results and assess the methods performance to help in validating the thesis hypothesis. We analyze the methods in therms of accuracy, processing power consumption and also network overhead. Specific SDN technologies were chosen in order to build an SDN infrastructure, as described in the next sections.

6.1 Experimental environment

For our experiments, a virtual network was configured on Mininet [Min18, LHM10], a platform that allows the creation and emulation of scalable virtual networks with controllers, switches, hosts, links, and also enables the development of different topologies through Python scripts. The Data plane was established using OpenvSwitch [PPK+09] virtual switches (OvS). On the Control plane, an OpenDaylight [OPE18b] controller was installed and configured on a remote computer. OpenFlow was chosen as SDN controller in our experiments due to its detailed documentation and as it supports java based implementations and provides REST interfaces. This is useful for our statistic collection process implementation. However, the methods proposed in this thesis can be used even when other SDN protocols and controllers are adopted, as discussed in Section 2.1. On the Application plane our protection methods are running, being METHOD_A and METHOD_B on the module called "Encrypted DDoS and port scan detection" and METHOD_C on the proposed module "Generic encrypted attacks detection".

The experiments were performed on an Apple Macbook Pro computer, with 16GB of RAM (2133MHz), an Intel Core i7 @ 2.8GHz CPU and 256GB SSD. On the Control plane, a Virtualbox [Ora18] machine with 4GB of RAM and one CPU core was configured on this hardware, with an Ubuntu 17.1 [Ltd18] Operating System. An Opendaylight Oxygen-SR1 [OPE18b] controller was installed on this virtual machine. This controller was used for testing and evaluating our proposed detection and protection methods. The Data plane was established using OpenvSwitch [PPK+09] virtual switches. As a Southbound communication protocol, we used OpenFlow version 1.4 [Foua] [Fou14]. OpenFlow was chosen because it provides the statistical data required by our methods and is the most used protocol in SDN nowadays, being also supported by most SDN controllers. Opendaylight controller was chosen due to its friendly interface, REST API communication provided, java and OpenFlow support. However, other protocols and controllers could be used in this evaluation.

6.2 Experimental evaluation on detecting port scan

In this section we present our port scan detection evaluation. First, port scans were performed in the experimental network implemented and then we analyze how METHOD_A identifies and reacts to such attacks.

6.2.1 Performing port scan

To perform some port scan attacks on our network, initially tests were generated to verify the possibility of detecting false positives, when normal requests are made to the servers. Therefore, a script that makes 200,000 requests to the different servers (see attacker and targets in Figure 6.1) was created and executed. The number of packets vary between 10 and 3,000, depending on the request. Hence, the system must allow its routing, because none of the flows has a number of packets equal to or less than five. By analyzing the generated logs and the blacklist, no entry of port scan attempt was found, showing that our system did not detect any false positive in such situations.

After that, port scanning flows were generated using Nmap [Lyo18] [Lyo09]. Initially, a same port number was scanned on different hosts, performing a horizontal scan on the network, using the “nmap -p PORT ADDRESS" command. The vertical scan test was performed using the “nmap -sT ADDRESS" command, which scans distinct ports on a target address. The last scan returned three open ports (22, 53 and 80).

This test allowed the attacker to enhance a large number of scans at a lower interval than the collection because it was defined on three seconds. However, after this interval, new attempts will be blocked because the sum of streams with a number of packets equal to or less than five will be greater than the established threshold for being considered a port scan.

Scanning tests were also executed without the SYN flag trough the “nmap -sf ADDRESS" command, which performs FIN scanning. In this case the controller has discarded the packets and the scanning did not materialize.

The experimental network topology used in our experiments is shown in Figure 6.1. It is composed by an OpenDaylight controller managing four switches. There are also five servers, running services, such as web servers and databases, and three hosts that generate normal requests to the servers. Also, a malicious host is included in this network, with IP 10.0.0.11, which is running our port scan attack script.

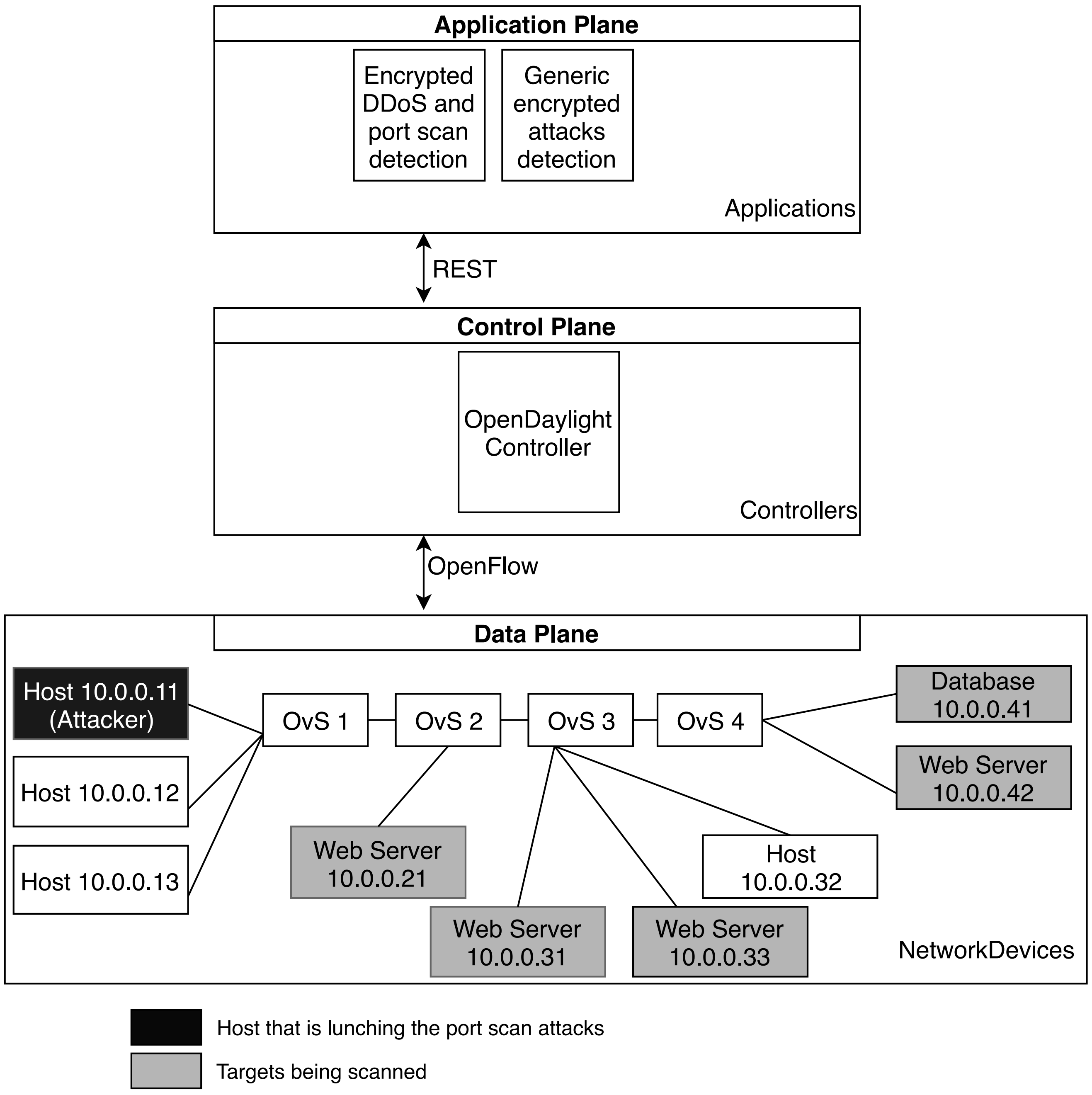

Figure 6.1 – Experimental network topology to evaluate port scan

6.2.2 Detecting and preventing port scan

Port scanning streams are typically limited to a maximum of three packets, for example TCP Connect has SYN, SYN/ACK and ACK packets, except in the cases that packets are re-transmitted. Furthermore, statistically, port scans occur at small time intervals. Since there is a periodical collection of flow statistics, it is possible to analyze whether there was an increase on the number of packets between two queries. Therefore, we consider that a stream is a port scan attempt when the number of packets is equal or less than five, by setting the threshold Scans_per_Host_Threshold to five. This number corresponds to the three packets responsible for the handshake of a TCP connection added with two tolerance packets, in case of re-transmission. Since this number of packets is very low, the probability of having a non-malicious flow detected as scanning is also very low. The number of target hosts (Target_Host_Threshold) needed to characterize the attack attempt was defined as three hosts due to the limited size of the network used for testing. Those thresholds are used in Algorithm 5.1 to perform horizontal port scan detection. To perform vertical scan detection, the common scanned ports weight (CSP_Weight) was set to five, while the weight for any other port (OSP_Weight) is three. Those values are required as shown in Equation 5.1. The ports that we chose to have a higher probability to being scanned are 22, 23, 25 and 3389, as they are the most scanned ports according to the statistics of incidents reported by CERT.br [CER18]. The detection threshold was set to 15. We observed that a higher threshold may reduce the method effectiveness, since an attacker may be able to perform more scans before being detected, and a lower threshold, could compromise legitimate access and present a high number of false positive detection.

Our method computes this data to perform detection. Table 6.1 shows flows stored in the collected counters database of a given host. The stored flow information is source and destination IP and ports, number of packets and an indication whether the entry corresponds to the request or to a response to the request that originated the flow.

In the example in Table 6.1, host 10.0.0.11 produced nine flows. Each flow has two entries on the table, indicating the destination and the request return. Through the column "Packets", a normal access behavior is shown in the first four flows, since the number of packets is greater than five. Lines 9 to 14 show that the number of packets of the respective flows is less than five, characterizing a port scan attack.

According to the defined rules, each port scan has a weight of five, and as the sum of these reaches the 15 threshold, the IPS considers these flows as an attack attempt, and discards them. After this, the attacker originated two more streams, which were discarded and information is stored in the database. This can be seen on the last two entries in Table 6.1. They show an amount of one packet per stream, which corresponds to the SYN packet received by the controller. The source and destiny addresses are represented in columns "SRC ADDRESS" and "DST ADDRESS" and the values are IP v4 addresses. The message type considered was "REQUEST" and "RESPONSE", which are represented by the values "REQ" and "RESP", respectively.

Table 6.2 presents a quantitative analysis of the flows. Non-malicious flows are shown on the second column. All 200,000 generated flows were forwarded, with no false negative alarms. The third column shows the horizontal port scan results. For this, 20 distinct flows were generated. Since the validation is performed after scanning three distinct hosts, the maximum number of malicious streams to be forwarded is 12 (three per target). However, some packages were re-transmitted and the IPS received both, original and re-transmitted packets, recognizing these as two distinct flows and, thus, the analysis resulted

Table 6.1 – Some flows information stored in the created database

ID	SRC ADDRESS	DST ADDRESS	SRC PORT	DST PORT	PACKETS	TYPE
1	10.0.0.11	10.0.0.21	33666	80	103	REQ
2	10.0.0.21	10.0.0.11	8080	33666	96	RESP
3	10.0.0.11	10.0.0.21	33668	22	65	REQ
4	10.0.0.21	10.0.0.11	22	33668	61	RESP
5	10.0.0.11	10.0.0.21	33670	110	68	REQ
6	10.0.0.21	10.0.0.11	110	33670	62	RESP
7	10.0.0.11	10.0.0.21	45985	110	68	REQ
8	10.0.0.21	10.0.0.11	110	45985	61	RESP
9	10.0.0.11	10.0.0.31	56974	23	2	REQ
10	10.0.0.31	10.0.0.11	8080	56974	1	RESP
11	10.0.0.11	10.0.0.31	26261	22	1	REQ
12	10.0.0.31	10.0.0.11	22	26261	1	RESP
13	10.0.0.11	10.0.0.31	56974	25	2	REQ
14	10.0.0.31	10.0.0.11	25	56974	1	RESP
15	10.0.0.11	10.0.0.31	48906	110	1	REQ
16	10.0.0.31	10.0.0.11	110	48906	0	RESP
17	10.0.0.11	10.0.0.31	27695	110	1	REQ
18	10.0.0.31	10.0.0.11	110	27695	0	RESP

in a number of 14 streams. These streams were now considered malicious only in the fourth or fifth checked host. These values can be seen in the third column of Table 6.2.

In the fourth column, which represents the execution of vertical scan, 1,000 malicious flows were generated. This generation, being automatic, was considerably faster than the collection interval, which resulted in the routing of 658 malicious flows. By the time our software terminated the analysis, they were considered as malicious source and were added to the blacklist and the system discarded the remaining 342 packets. In addition, this port scan also had some duplicated packets, causing 12 malicious streams that were considered as non-malicious streams.

The last column shows the number of FIN exploitation scans. As explained previously, uninitialized streams with SYN packets are immediately discarded. Thus, all originated flows were discarded.

Table 6.2 – Port scan test result with an empty blacklist

Flows	Not scan	Hor. scan	Vert. scan	FIN scan
Gernerated	200000	20	1000	1000
Forwarded	200000	14	658	0
Blocked	0	8	342	1000
False negatives	0	2	12	0

The same procedure was performed again to check if all streams were indeed added to the blacklist. In this procedure the blacklist has not been cleared and the result is shown in Table 6.3. As it can be observed, all generated flows, including non-malicious ones, were discarded. Similarly to the previous test, at some point the hosts performed a port scan, then they were added to the blacklist, and therefore were blocked when trying to perform new attempts to execute new accesses to the host. It is important to notice that both false positives and false negatives were not observed in this scenario. The average flows delay imposed by our IPS was five milliseconds, due to blacklist check and controller rules addition.

Table 6.3 – Port scan test result when the blacklist contains entries

Flows	**Not scan**	**Hor. scan**	**Vert. scan**	**FIN scan**
Gernerated	200000	20	1000	1000
Forwarded	0	0	0	0
Blocked	200000	20	1000	1000

6.3 Experimental evaluation on detecting DoS

As previously discussed, an attacker may be able to perform DoS attacks in several ways, for example, by sending a large number of encrypted messages through the encrypted channel to empower the attacks due to the encryption and decryption process. By using OpenFlow protocol, for example, different messages can be used to perform such attacks, such as OFPT_FEATURES_REPLY, OFPT_PORT_STATUS, OFPT_PACKET_IN, OFPT_PACKET_OUT and OFPT_FLOW_MOD.

6.3.1 Performing DoS

To validate METHOD_B, DoS attacks were also performed in our network. We developed scripts that generate network traffic, being generated normal and malicious flows. The script that generates DoS attack was ran on three different hosts at the same time (IPs 10.0.0.11, 10.0.0.12 and 10.0.0.13) that generate bogus messages through the encrypted channel. These messages were composed by OFPT_PORT_STATUS, OFPT_PACKET_IN, OFPT_PACKET_OUT and OFPT_FLOW_MOD, performing a DoS attack against the switch and also against the controller. It is important to note, that by overloading the switches and the controller, the servers that provide services to the network may be not reached and thus those services may also be not accessible under the DoS attacks being simulated in this situation.

The experimental network topology used in our experiments to simulate DoS attacks is shown in Figure 6.2. It is composed by an OpenDaylight controller managing four

OpenVSwitch switches. There are also five servers, running services, such as web servers and databases, and three hosts that are lunching the DoS attacks.

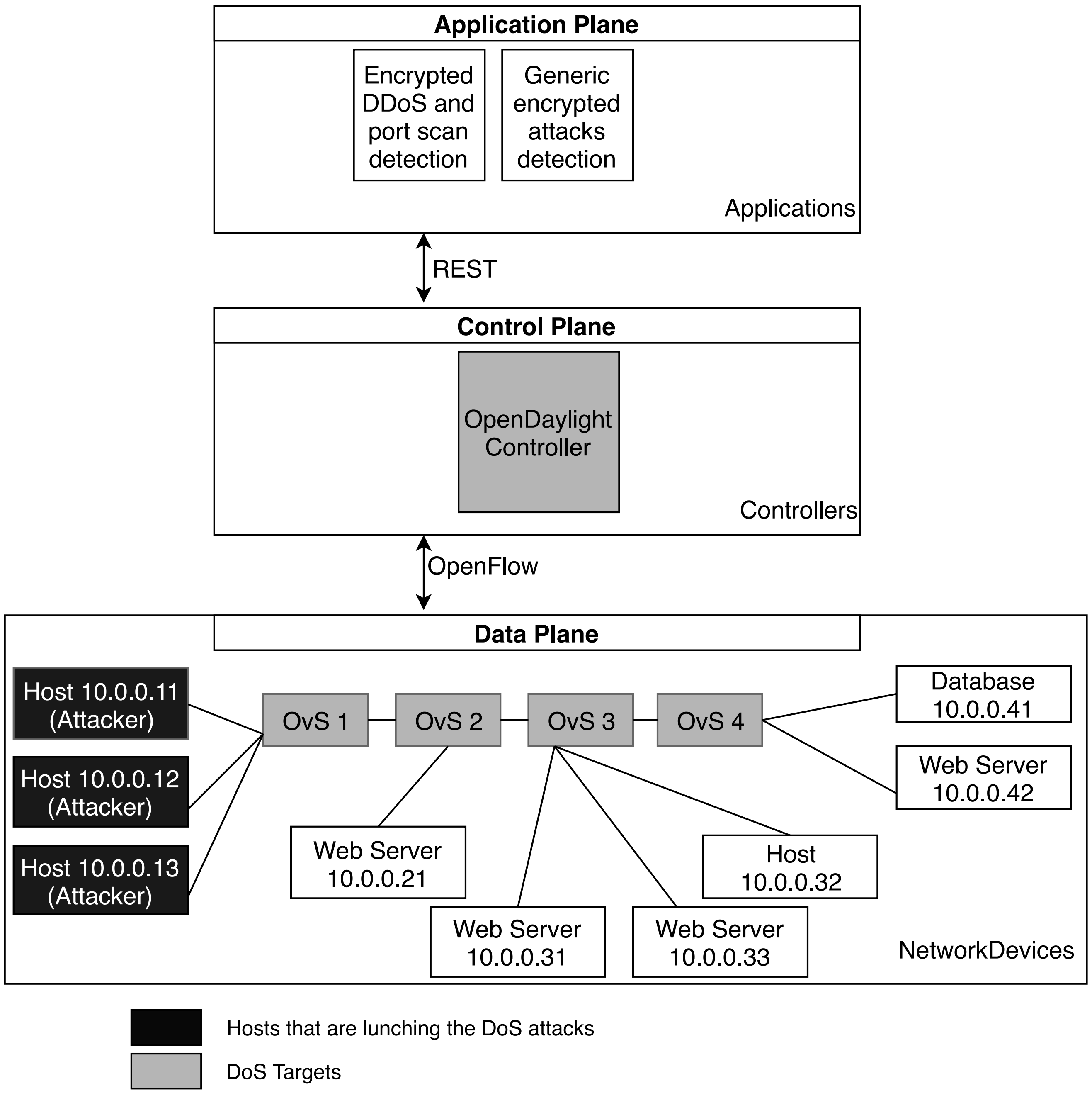

Figure 6.2 – Experimental network topology to evaluate DoS

Openflow provides secure communication beteween switch and controller by using TLS over its encrypted channel. Thus, a TLS connection was initiated by the switch on startup to the controller, which was listening on the TCP port 6653 (this is the default port, which can also be changed as required). After, switch and controller exchanged certificates signed by a site-specific private key (provided by Opendaylight) to perform authentication. Each switch is configured with one certificate for authenticating the controller (controller certificate, i.e., controller public certificate) and the other for authenticating to the controller (switch certificate, i.e., switch private certificate). A script was executed to collect and store statistics from the switches every three seconds (same as described in Section 6.2). Two different situations were simulated. In each simulation, the following data were generated:

- DoS attack: 20,000 packets per second from a compromised host to the switch with an invalid destiny.
- Normal traffic: 4,000 packets per second being normal traffic exchange between legitimate hosts (files being transferred).

DoS Simulation 1. In this simulation, normal traffic was injected first, during nine seconds. After, DoS traffic was injected during the remaining simulation time (the entyre simulation time was 30 seconds). Thus, the first samples are based on normal traffic only and the following samples contain both normal and DoS traffic.

DoS Simulation 2. In this simulation, normal traffic was first injected. After nine seconds, the attacking script was initiated during six seconds and stopped again, while the normal traffic injection was still being executed. Thus, the first nine collected samples in this simulation, are based on normal traffic only, the follow three samples are from normal traffic and also attacking flows, and the last five samples in this simulation are based on normal traffic only.

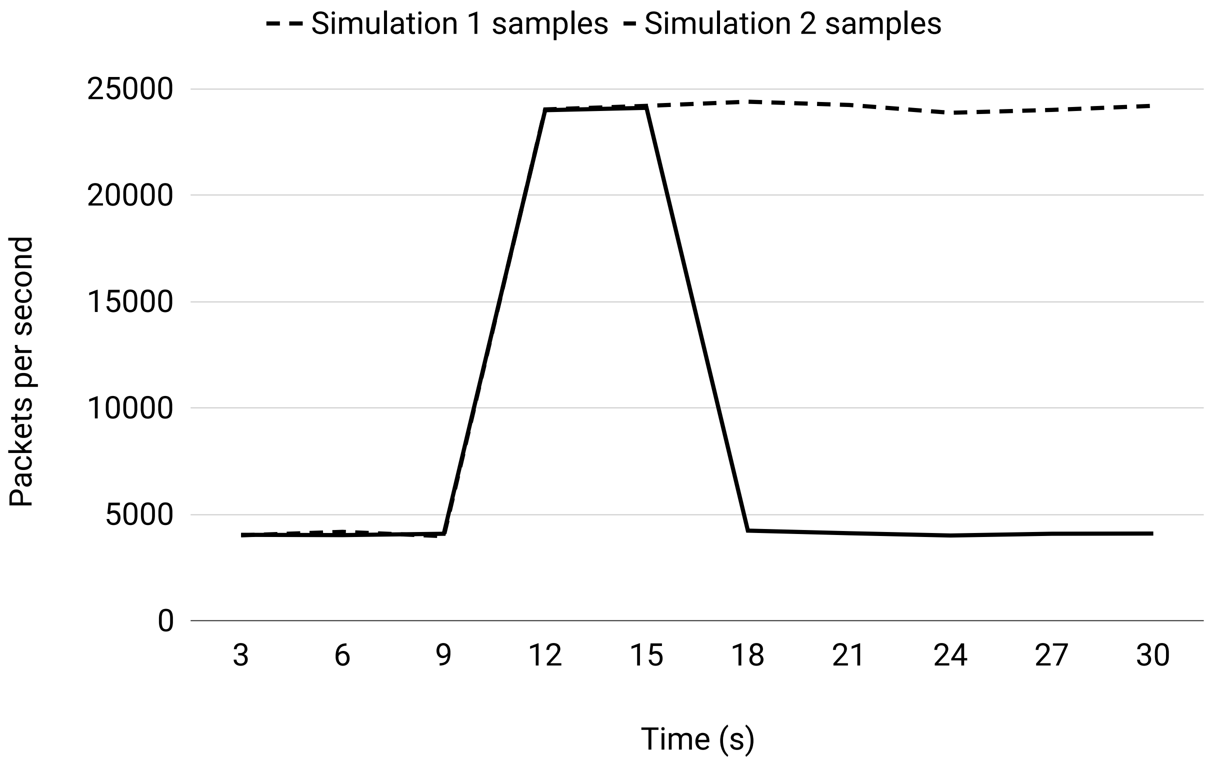

Figure 6.3 – Packets per second generated in simulations

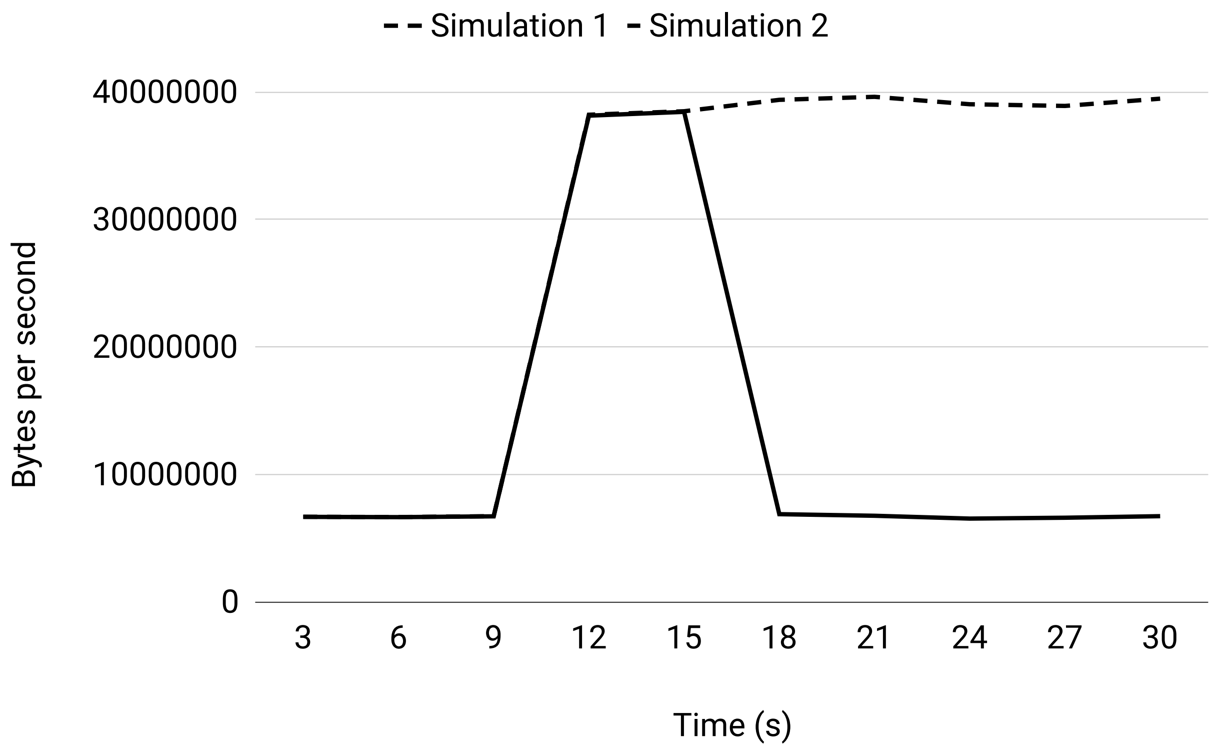

Figure 6.4 – Bytes per second generated in simulations

Figure 6.3 and Figure 6.4 show the number of generated packets per second and bytes per second in both simulations. In DoS Simulation 2, a burst increase in both observed features is shown at sample 4 (time 12 s), being decreased to previous levels at sample 6 (time 18 s). However, in DoS Simulation 1, similar burst increase is shown at sample 4 (time 12 s) with no significant decrease during the remaining simulation. Thus, the second simulation could represent normal traffic with some noise or failure from samples 4 to 6 (times 12 to 18), while the first simulation could represent an encrypted DoS. Those situations were simulated in our experiments as they are very common in real networks.

In those simulations, both packet-per-second and byte-per-second features had a similar behavior. However, an attacker could be able to perform similar attacks by using a variety of packet length, or packet-per-second. Therefore, our method is using both features to improve protection.

6.3.2 Detecting and preventing DoS

First, the required thresholds for the detection methods were established. Those values were defined based on several experiments, as their results were considered more accurate and with less false positive alerts.

- Thresholds used to compute bytes per second:
  - C_Threshold_BS: 3%
  - Max_Anomaly_Samples_BS: 40%
  - Time_Interval: 30 seconds
  - D_Threshold_BS: 5%
- Thresholds used to compute packets per second:
  - C_Threshold_PS: 3%
  - Max_Anomaly_Samples_PS: 40%
  - Time_Interval: 30 seconds
  - D_Threshold_PS: 5%

Those thresholds and the stored switch counters were then used as input for our METHOD_B, so that the samples could be analyzed to perform detection. This detection method uses two different analysis strategies, as described in Section 5.3.2. The first detection strategy is based on our customized CUSUM and compare the computed values to a threshold regarding the samples average, and the second strategy computes the difference between the samples and the average to compare it to the defined threshold for considering a sample as normal or abnormal. The OpenFlow per port counters *Received/Transmitted packets and Received/Transmitted* bytes where collected in the previous stage and are used as input for this detection stage.

Results in Simulation 1

The collected samples values and also the computed values that are used to perform detection are presented in the following tables. The values in columns *i* and *j* are only an index that is incremented on each sample, being *i* used for the samples that are being analyzed and *j* an index that is used for the samples that were considered normal ones. The column *S* presents the samples values and column *Avg(i)* has the samples average, which is computed on the samples that have already been analyzed and identified as normal behavior and the sample being analyzed. Therefore, column *S_Aux(j)* presents the values of the samples that were considered normal values, because if the samples average is computed over the whole samples set, our method will fail on detection if only one (or a few) abnormal sample is found. In column *C(i-1)* the cumulative sum values of previous sample is show and in *C(i)* the value of the cumulative sum considering the sample being analyzed is show. In column *Threshold* the values computed by multiplying the samples average value and the defined thresholds (C_Threshold_PS, C_Threshold_BS, D_Threshold_PS and D_Threshold_BS), which are represented in column *Threshold*.

The values computed on the first ten samples from both simulations by our customized CUSUM are show in Table 6.4 and in Table 6.5. First, the average of normal samples is computed. Then, the cumulative sums process starts. The values of the cumulative sums are show in column C(i) and the previous values in column C(i-1). After, the threshold is computed (3% of the average, as previously defined for our experiments). Finally, detection is performed. If an abnormal sample is found, *i.e.*, if the value in column *C(i)* is above the value in *Threshold*, a counter is incremented in column *Attack*. Our METHOD_B defines that an attack was found if the end value in column *Attack* is equal to or above 4 (the threshold of abnormal samples defined in *Max_Anomaly_Samples_PS* and *Max_Anomaly_Samples_BS* was defined as 40% of the analyzed samples, with represents 4 samples in our experiments). The *Attack* end values are 7, as 7 abnormal samples were found in each detection rule, which exceeds the abnormal ones threshold (4). Therefore, an attack is detected by both rules.

Table 6.4 – DoS detection in Simulation 1 with CUSUM on PS

i	**S(i)**	**Avg(i)**	**j**	**S_Aux(j)**	**C(i-1)**	**C(i)**	**Threshold**	**Attack**
1	4019	4019	1	4019	0	0	120.57	0
2	4182	4100.5	2	4182	0	81.50	123.01	0
3	3979	4060.00	3	3979	81.50	0.50	121.80	0
4	24028	9052.00			0.50	14976.50	271.56	1
5	24196	9094.00			14976.50	30078.50	272.82	2
6	24391	9142.74			30078.50	45326.75	274.28	3
7	24238	9104.50			45326.75	60460.75	273.13	4
8	23871	9012.75			60460.25	75318.50	270.38	5
9	24006	9046.50			75318.50	90278.00	271.39	6
10	24198	9094.50			90278.00	105381.50	272.83	7

Table 6.5 – DoS detection in Simulation 1 with CUSUM on BS

i	**S(i)**	**Avg(i)**	**j**	**S_Aux(j)**	**C(i-1)**	**C(i)**	**Threshold**	**Attack**
1	6677384	6677384	1	6677384	0	0	200321	0
2	6651879	6664631	2	6651879	0	-12752	199938	0
3	6723156	6684139	3	6723156	-12752	26263	200524	0
4	38219256	14567918			26263	23677601	437037	1
5	38492491	14636227			23677601	47533864	439086	2
6	39388375	14860198			47533864	72062041	445805	3
7	39624231	14919162			72062041	96767109	447574	4
8	39041886	14773576			96767109	121035419	443207	5
9	38910195	14740653			121035419	145204960	442219	6
10	39479076	14882873			145204960	169801163	446486	7

Follow, Table 6.6 and Table 6.7 show the results when differences are computed with the average to perform detection. The values in column *i* are only an index that is incremented on each sample. The column *S* presents the samples values and column *Avg* has the samples average, which is computed on the samples that have already been analyzed and identified as normal behavior and the sample being analyzed. Therefore, column *S_Aux(j)* presents the values of the samples that were considered normal values, because if the samples average is computed over the whole samples set, our method will fail on detection if only one (or a few) abnormal sample is found. Column *S(i) - Avg(i)* has the value of the difference between the sample being analyzed and the average of normal samples and the sample being analyzed. In column *Threshold* the values computed by multiplying the samples average and the defined threshold (D_Threshold_PS and D_Threshold_BS, defined as 5% of samples average value). If an abnormal sample is found, *i.e.*, if the value in column *S(i) - Avg(i)* is above the values in *Threshold*, a counter is incremented in column *Attack*. Finally, our method defines that an attack was found if the end value in column *Attack* is equal to or above 4 (the threshold of abnormal samples defined in *Max_Anomaly_Samples_PS* and *Max_Anomaly_Samples_BS* was defined as 40% of the analyzed samples).

Table 6.6 – DoS detection in Simulation 1 with difference analysis on PS

i	S(i)	j	S_Aux(j)	Avg(i)	S(i) - Avg(i)	Treshold	Attack
1	4019	1	4019	4019.00	0	200.95	0
2	4182	2	4182	4100.50	81.5	205.05	0
3	3979	3	3979	4060.00	-81	203.00	0
4	24028			9052.00	14976	452.60	1
5	24196			9094.00	12115.2	604.04	2
6	24391			9142.75	10258.5	706.62	3
7	24238			9104.50	8661.85	778.80	4
8	23871			9012.75	7258	830.00	5
9	24006			9046.50	6571.5	871.72	6
10	24198			9094.50	6087.2	905.54	7

Table 6.7 – DoS detection in Simulation 1 with difference analysis on BS

i	S(i)	Avg(i)	j	S_Aux(j)	S(i)-Avg(i)	Threshold	Attack
1	6677384	6677384	1	6677384	0	333869	0
2	6651879	6664631	2	6651879	-12752	333231	0
3	6723156	6684139	3	6723156	39016	334206	0
4	38219256	14567918			23651337	728395	1
5	38492491	14636227			23856263	731811	2
6	39388375	14860198			24528176	743009	3
7	39624231	14919162			24705068	745958	4
8	39041886	14773576			24268309	738678	5
9	38910195	14740653			24169541	737032	6
10	39479076	14882873			24596202	744143	7

Results in Simulation 2

In the second simulation METHOD_B both detection rules were also analyzed. First, Table 6.8 and Table 6.9 present the results when our customized CUSUM is applied. Follow, in Table 6.10 and in Table 6.11 the results of the difference analysis are presented. The *Attack* end value is 2 in the four tables, as only 2 abnormal samples were found in each simulation. The threshold for abnormal samples was also defined as 40% of the analyzed samples, with represents 4 samples. As only 2 analyzed samples exceed the abnormal ones threshold (4), an attack is not detected in Simulation 2. This strategy avoids a large number of false positive alarms, only one (or a few) samples may not represent an attack, due to some noise or network failure, for example. Therefore, our method discards the abnormal computed samples by storing only the normal computed samples in a new data structure *S_Aux(j)* to compute the remaining samples correctly by using our customized CUSUM and also the difference value between samples and average.

Table 6.8 – DoS detection in Simulation 2 with CUSUM on PS

i	**S(i)**	**Avg(i)**	**j**	**S_Aux(j)**	**C(i-1)**	**C(i)**	**Threshold**	**Attack**
1	4042	4042.00	1	4042	0	0	121.26	0
2	4029	4035.50	2	4029	0	-6.50	121.06	0
3	4092	4054.33	3	4092	-6.50	31.16	121.63	0
4	23992	9038.75	4	4239	31.16	14984.42	271.16	1
5	24103	9066.50	5	4117	31.16	15067.67	271.99	2
6	4239	4100.50	6	4011	31.16	169.66	123.01	2
7	4117	4103.80	7	4093	169.66	182.86	123.11	2
8	4011	4088.33	8	4102	182.86	105.53	122.65	2
9	4093	4088.00			105.53	109.53	122.67	2
10	4102	4090.62			109.53	120.90	122.71	2

Table 6.9 – DoS detection in Simulation 2 with CUSUM on BS

i	**S(i)**	**S_Aux(j)**	**j**	**Avg(i)**	**C(i-1)**	**C(i)**	**Threshold**	**Attack**
1	6669300	6669300	1	6669300	0	0	200079	0
2	6647850	6647850	2	6658575	0	-10725	199757	0
3	6710880	6710880	3	6676010	-10725	24145	200280	0
4	38147280	6879897		14543827	24145	23627597	436314	1
5	38444285	6755997		14618078	24145	23850351	438542	2
6	6879897	6533919		6726981	24145	177969	201809	2
7	6755997	6602009		6732784	177060	200272	201983	2
8	6533919	6723178		6699640	200272	34550	200989	2
9	6602009			6685693	-49133	-49133	200570	2
10	6723178			6690378	-16333	-16333	200711	2

Table 6.10 – DoS detection in Simulation 2 with difference analysis on PS

i	S(i)	j	S_Aux(j)	Avg(i)	S(i) - Avg(i)	Threshold	Attack
1	4042	1	4042	4042.00	0	202.10	0
2	4029	2	4029	4035.50	-6.5	201.77	0
3	4092	3	4092	4054.33	32	202.71	0
4	23992	4	4239	9038.75	14940	451.93	1
5	24103	5	4117	9066.50	12022.2	453.32	2
6	4239	6	4011	4100.50	-6510.5	205.02	2
7	4117	7	4093	4103.8	-5685	205.19	2
8	4011	8	4102	4088.33	-5067.1	204.41	2
9	4093			4089.00	-4431.2	204.45	2
10	4102			4090.62	-3980	204.53	2

Table 6.11 – DoS detection in Simulation 2 with difference analysis on BS

i	S(i)	S_Aux(j)	j	Avg(i)	S(i)-Avg(i)	Threshold	Attack
1	6669300	6669300	1	6669300	0	333465	0
2	6647850	6647850	2	6658575	-10725	332928	0
3	6710880	6710880	3	6676010	34870	333800	0
4	38147280	6879897		14543827	23603452	727191	1
5	38444285	6755997		14618078	23826206	730903	2
6	6879897	6533919		6726981	152915	336349	2
7	6755997	6602009		6732784	23212	336639	2
8	6533919	6723178		6699640	-165721	334982	2
9	6602009			6685693	-83684	334284	2
10	6723178			6690378	32799	334518	2

6.4 Network and hardware overhead

In order to measure hardware consumption, we injected some network traffic (only normal file transfers and control messages, without injecting malicious data) and observed the controller CPU time and memory consumption without activating our IPS. After that, our IPS was activated and the same traffic was injected. We then computed the difference on hardware consumption to determine the amount of resources that our IPS uses, first considering that only METHOD_A was activated and then only METHOD_B was activated. When the controller was running without the IPS, it consumes 1.8% CPU and 29% RAM. By activating the IPS and METHOD_A, the resource usage increases to 3.1% CPU and 32.2% RAM. Thus, it consumes 1.3% of CPU and 3.2% of RAM from the used virtual machine. After, only METHOD_B was activated and new measures were made. The resource usage increases to 3.3% CPU and 32.9% RAM. Thus, it consumes 1.5% of CPU and 3.9% of RAM from the used virtual machine (see Section 6.1). Those values were computed based on the average of 30 samples collected on a 180 s simulation time (a measure was made each 3 seconds). Table 6.12 summarizes the CPU and memory usage by the developed methods. Those values were computed based on 17 repetitions, considering a 95% confidence interval.

Table 6.12 – CPU and Memory overhead

	CPU (%)	**Memory (%)**
METHOD_A	1.3	3.2
METHOD_B	1.5	3.9

Figure 6.5 shows the average of CPU usage on each of the 17 repetitions without the IPS and also the increase in CPU usage by both evaluated methods. Figure 6.6 shows the average of RAM usage by our IPS on each of the 17 repetitions.

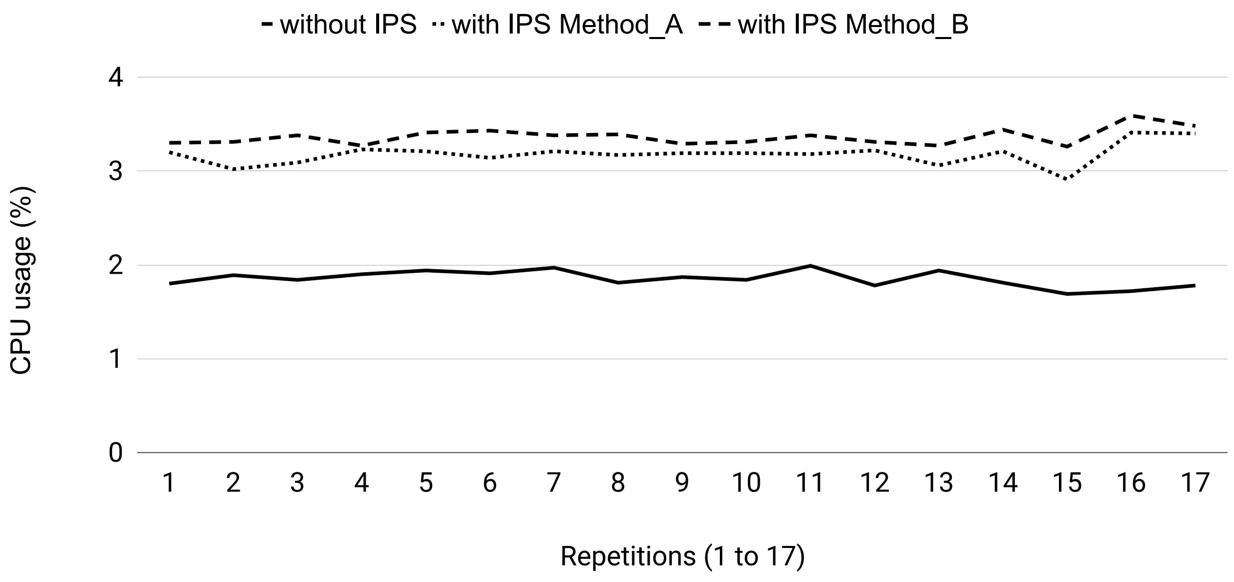

Figure 6.5 – CPU usage

The following message types were used between our IPS and the switches. The overhead caused by those messages was measured to verify how many packets our solutions would produce in the network compared to the total packets that were produced in our experiments. These results were included in the computation of the average network bandwidth usage.

- OpenFlow_v4.multipart_reply.type == OFPMP_FLOW
- OpenFlow_v4.multipart_request.type == OFPMP_FLOW

Those messages are responsible for around 0.13% of the whole packets number transmitted on the network. The consumed bandwidth by these packets on the experimental network was around 0.0038%. These results were obtained through scenarios composed by the proposed IPS in a network topology presented in Figure 6.2 and transfers of 3.3GB

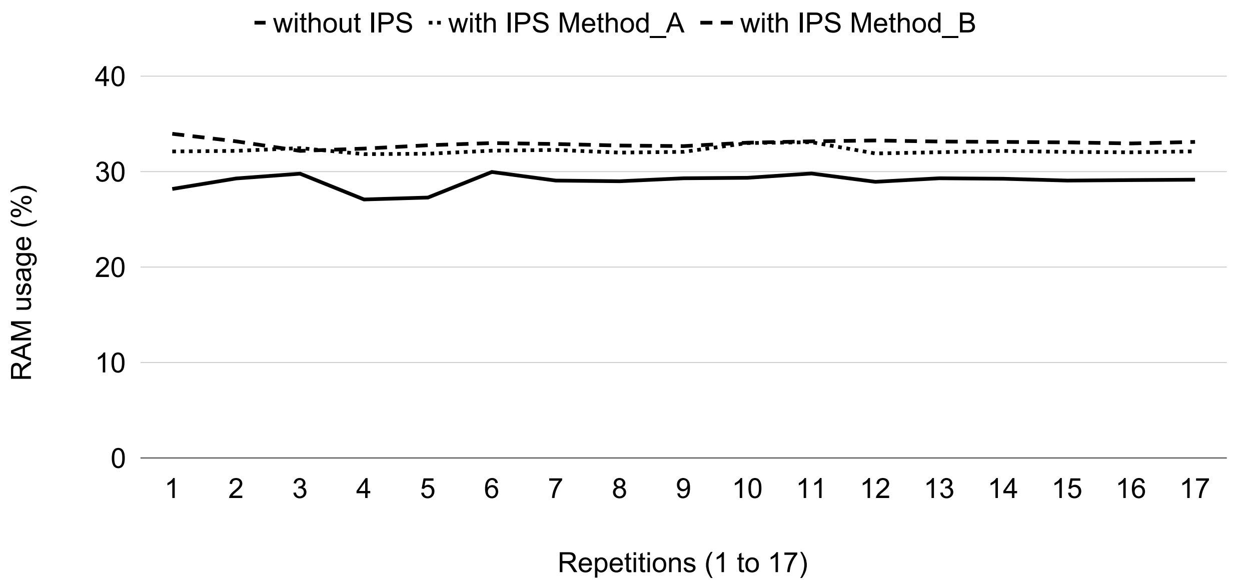

Figure 6.6 – RAM usage

files between the servers, to simulate normal network traffic. On those experiments, 646,691 packets, in a total of 10,008,388,120 bytes, were transferred in the network. Our proposed IPS produced 837 packets, in a total of 383,292 bytes, of network traffic. These experiments were repeated 17 times to achieve a confidence level of 95%. It is important to notice that this bandwidth usage changes according to the network traffic.

6.5 General discussion

The proposed solution deals with the starting point of several different types of attacks, for example, port scan and DoS (already mentioned and detected as shown in this chapter). As mentioned before, usually an attacker first scan some ports in a host (or on several hosts) prior to an actual attack. Chapter 5 presented a solution to detect port scans that may be used to find possible vulnerabilities points prior to an attack. One of such attacks, and very common lately, is a Denial of Service (or Distributed Denial of Service) attack. Typically, the purpose of DoS attacks is to consume resources, such as the network bandwidth, CPU and memory of a given equipment, like an SDN switch, SDN controller or application, in order to deny service to legitimate users or even to affect it so that it may not work as it is intended to [ZHKS16].

A DoS attack may be more efficient when encrypted packets are used, as it consumes more hardware resources to being processed, according to the packets type [ZHKS16, Gar]. Furthermore, the increasing grow of encrypted traffic, specially SSL/TLS, on the Internet, as well as on enterprise networks, on the one hand provides privacy and authentication, but on the other hand compromises threat detection. Current protection systems and organization's security controls, like IDS/IPS, Next Generation Firewalls, Unified Threat Management, Secure Gateways, Data Loss Prevention and anti-malware solutions, must be improved so that they may be able to combat this new attackers strategy, since they may be not working on more than 50% of new network attack methods currently [Gar].

A plethora of other attacks, like insider attacks (from malicious or compromised insider), including frauds, sensitive data evasion and industrial espionage are being improved by attackers, specially by the addition of encryption on malicious packets, and must be dealt with on SDN by security systems, like IDS/IPS. The main limitation of current SDN IDS, found in the literature, is their inability of detecting encrypted threats, as packet analysis cannot be performed since decryption can violate ethical norms and regulations on privacy. Therefore it is necessary to perform network traffic analyzes without packets decryption to detect encrypted threats.

The above mentioned threats can be dealt with using the solution presented in Chapter 5. By analyzing switch counters and then dropping the malicious flows before they start an attack (or reacting to an attack very fast) can be achieved by our solution. This is realized since attacks that use port scan will be detected before they even start (as shown in Chapter 5).

The hypotheses defined to this thesis was validated in this research according to our detection methods and results, as shown in Chapters 5 and 6. We also show what kind of methods and approaches found in the literature could be used and adapted to detect such attacks. In short, we show that SDN switch counters data can be used to detect attacks in SDN, even when packets are encrypted and sent through an encrypted channel. This approach also does not affect significantly network performance and is lightweight in terms of hardware resource usage. The proposed methods are lightweight in therms of:

- Network usage: our methods are based on statistical data that is collected through specific messages each three seconds. This traffic is around 0.13% from the whole traffic in our experiments.

- CPU usage: as only a few computations are necessary on the stored statistical data, our detection methods CPU consumption is around 1.4% for METHOD_ A and 1.5% for METHOD_B.

- Memory usage: the memory usage depends on the number of connected devices in the network. In our experiments, the developed IPS used 3% of the available RAM on the controller.

- Flows forwarding delay: a delay average of 5 milliseconds was observed in our methods on analyzing flows, due to the rules matching and definition time and also to check if the incoming flow source is in the blacklist.

To support this thesis hypotheses validation, our four research questions could be answered as:

- RQ1: What are the main attacks in SDN? In Section 2.3 we described several attacks that traditional networks are vulnerable too, but that also could be adapted to SDN. Besides, we also discuss attacks that may be performed specifically in SDN. In Section 5 we also described how attacks could affect SDN networks according to the STRIDE model. We also found that encryption could be used to empower such attacks, specially denial of service, due to encryption and decryption process when a large number of bogus encrypted messages are sent through the encrypted communication channel between switch and controller.

- RQ2: Which limitations current IDS have on detecting attacks in SDN? The main limitations found, according to our SMS and other research, is the inability of current methods to analyze encrypted data, using packet-based approaches. Another limitation, is the overhead that is typically added to the network by such protection systems, besides the high false positive rates and high hardware resource consumption needed by most of the methods that were analyzed. Furthermore, we found that encrypted attacks may be detected by using three main approaches: protocol-based, modification-based and statistical-based, as discussed in Chapter 5.

- RQ3: How to protect SDN against malicious encrypted traffic? In Section 5 we described our approach to detect such issues in SDN. As packet inspection is difficult and not feasible in terms of performance in encrypted packets, methods based on statistics and probability are most appropriate to be used for detecting anomalies in the network, both when packets are encrypted or not. Thus, methods to identify port scan and denial of service encrypted attacks are presented. Although those methods are intended to encrypted attacks, they could also be used to detect anomalies in non-encrypted traffic, as only statistical information is used. In specific networks, like internal sensor networks in an industry environment, a normal behavior (standard operation) can also be defined by using machine learning algorithms as a first step. As a second step, detection can be performed by searching for deviations of this standard operation, also by using machine learning algorithms, such as Naive Bayes and Random Forest (other algorithms can be applied in these steps as required). Therefore, traffic classification methods must also be used to group packets according to the protocol (*e.g.* application protocol). Thus, before those two steps are executed, the environment in which our methods will be applied to must be set up. One of the activities when setting up the environment is to define what are the protocols that are allowed in the network. This will allow new forwarding rules to be created or rules to drop packets from not allowed protocol will be inserted in the Data plane switches.

In our experiments, we consider 10 samples only to simplify the evaluations that are shown on the tables presented in this section (Tables 6.4 to 6.11). A larger set of samples was also evaluated, being composed by 30 samples collected in each simulation, according to the Central Limit Theorem [Pin17] that requires at least 30 samples. However, there was not found significant changes on the results in comparison to the smaller number of samples (10) that was used to describe the evaluations presented in this section, because our data was collected in a controlled environment (synthetic data) and is normalized. It is important to note, that larger data sets should be use on real networks to evaluate our methods, as we are intending to do as a next step.

6.6 Chapter summary

This chapter described some results from the detection and protection approaches being proposed. First, a simulated environment was build on Mininet, with OpenvSwitches, an Opendaylight controller and some hosts (See Figure 6.1). Then, a script was used to generate malicious port scan flows to evaluate our protection method (METHOD_A) intended to defend against such attacks. Our results show that protection was provided by first identifying the scans and then writing an entry in the blacklist and creating drop rules for such flows, based on its IP address.

After, DoS attacks were simulated on a similar environment (see Figure 6.2) to evaluate our METHOD_B. Another script was developed and used to inject bogus OpenFlow messages, such as OFPT_PORT_STATUS, OFPT_PACKET_IN/OUT and OFPT_FLOW_MOD, through the encrypted channel between the switches and controller, performing a DoS attack against the switch and also against the controller. Two different situations were simulated, being the first one initiated with normal traffic only (file transfers from one host to another) during 9 seconds. Then, the attacking script was started during 21 seconds. In the second simulation, the script that performs a distributed denial of service attack was started after 9 seconds that our simulation was initiated, during 6 seconds. Thus, in simulation two no attack was detected, as only 2 abnormal samples were found and a threshold for abnormal samples to detect an attack was set to 40% (4 samples). This threshold is important to avoid false positive alerts, that could be caused by network related problems. We show that the two different strategies used in METHOD_B, being the first one based on a customized CUSUM and the second one based on differences computation over the collected packet-per-second and bytes-per-second samples. It is important to note, that it is important to set up the required thresholds correctly according to the network on which METHOD_B will be used.

METHOD_C was also proposed in Chapter 5 to protect against other encrypted attacks, such as those lunched by an insider, like sabotage and data evasion. This approach could be implemented by using ISCX2012 and CIDIS2017 intrusion detection datasets to first define a normal behavior, as these datasets provide different files that have normal traffic only and also files that have malicious and normal traffic in. After, anomaly detection on the malicious packets stored in those datasets could be performed, both by using Random Forest, J48 and Naive Bayes algorithms. This strategy could be to classify malicious traffic according these 3 different algorithms, *i.e.*, at least two of them classify a given traffic as malicious, then an attack can be detected. A set of allowed protocols could be FTP, SNMP, HTTPS and DNS, as such traffic is available in the datasets. Furthermore, specific counters could be implemented on the switches flow tables by using P4, *i.e.*, the Data Plane could be customized on a way that traffic classification can be not flow based only but also based on packet information, so that the encrypted protocol can be identified more effectively. Thus, this method could be customized and used to identify other encrypted attacks than the possibilities presented in Chapter 5, such as data sabotage performed by an insider, that can be made against applications. This strategy could also be used in traditional IP networks based on packet information, such as inter-arrival-time and packet length.

Through the performed tests, the following observations were made on METHOD_A and METHOD_B, *i.e.*, on detecting port scan and encrypted DoS:

- Average time delay per flow of 5 milliseconds. Due to the need to check the blacklist and to add rules by the controller;
- 0.034 % of false negatives in detecting port scan. Due to the number of re-transmitted packets. Therefore, what should have been an attack ended up being considered a normal flow;
- Some scans carried out until its block. Although the detection algorithm was assertive in detecting scans, the interval between one analysis and another allowed the attacker to perform a few scans, being still necessary to use protection against other attacks, such as DoS.
- The networks bandwidth usage was around 0.0038% of the whole traffic in our simulations.
- The average RAM usage of the IPS on controller was 3%.
- After DoS and port scan were detected, drop rules were defined and entries in the blacklist were written so that new attempts from same IP source could not materialize. In the case of a legitimate host that is compromised, it will also be blocked in the network until a network's administrator procedure.

Thus, based on the presented results, we show that the attacks simulated in our environment were detected successful by our methods, using the thresholds that have been configured. We show that these methods are effective on protecting from the main attacks that can be performed using encryption against SDN switch, controller and applications, which are information disclosure and denial of service. We also show that these methods are lightweight, with low resources usage, such as CPU, memory and network bandwidth. However, our evaluations are based on artificial environment and simulated data, as SDN is a new paradigm in network communications and no IDS dataset that is suitable to evaluate our methods was found until this thesis was written. Finally, other overheads, such as the IPS database memory and CPU usage where not yet evaluated, as we only consider the resources usage by the IPS itself in this work.

7. CONCLUSION

In this research we discussed the rise adoption of encryption in computer networks communication. On the one hand, encryption provides more security, like increasing privacy and authentication on data exchange. On the other hand, attackers also are using encryption to empower methods used to malicious purposes and also to bypass protection systems, since the packets are encrypted and cannot be analyzed by typical packet-based methods and systems. However, we discussed in Chapter 5 that methods based on protocol analysis, modification analysis and statistical analysis can be developed based on metadata provided in the network. Such methods can act as an advanced security sensor to analyze encrypted traffic in order to identify encrypted threats and anomalies. Due to traditional networks ossification, new network technologies are emerging to fulfill the rising usage of computer networks nowadays. One such technology is SDN, which is managed by a centralized entity, called controller, and can use protocols, like OpenFlow to provide features and resources that may be used for detecting anomalies in network traffic. One such feature is the switch counters data stored in a flow table, that provides, for example, packet-per-second and byte-per-second counters (see Section 2.1), which can be used to provide network visibility in encrypted traffic and thus classify encrypted malicious flows. Such metadata can be used to model a normal traffic behavior and then search for abnormal behavior on incoming traffic, both by using, for example, machine learning algorithms (*e.g.* Random Forests and Naive Bayes Networks). Traffic classification methods can also be applied to identify, for example, the application protocol, and then apply network rules, such as to define what protocols are allowed on the network and also the amount of traffic regarding this protocol. Another way to analyze encrypted data is based on protocol analysis, by first identifying which protocol is used in data communication and then perform analysis based on its specification, such as packets length, inter-arrival time and packets or bytes per second.

In Section 5.1 we discuss the main attacks that SDN is vulnerable too, also including vulnerabilities that an attacker could explore by using encryption. Our research indicate that specially denial of service attacks, which may be preceded by an information disclosure attack (e.g. port scan), could be performed with encrypted data through the encrypted channels to empower it malicious purpose, as the encryption and decryption process requires more resources to be processed. Those attacks could be lunched against an SDN switch, controller and even applications. Thus, this thesis presents methods for detecting and also protecting against such attacks in SDN. Our methods are statistical-based on the switch counters data and intended specially to protect against port scan and DoS, although protection against other attacks can be provided by adapting our proposed methods. These data are available to the controller and may be collected through specific messages on predefined time intervals to perform detection and then to block threats. Furthermore, it does not overload the network and uses low processing power and memory. Thus, whenever an attacker performs a scan on the network, our IPS will detect it and then prevent new malicious attempts, like encrypted DoS and other application threats, improving security of SDN infrastructures. However, an attacker may not performs an scan in a target SDN or could be able to bypass our port scan protection method. Therefore, we also developed a method to identify denial of service in SDN, even if the flows are encrypted and the attack is lunched distributed. This method is also based on switch counters, which are packet-per-second and bytes-per-second, by using two different strategies, being: computing cumulative sums, i.e., a customized CUSUM, and also by computing the differences in a sample being analyzed and the average on normal samples analyzed.

We also presented a method for detecting other encrypted attacks, such as those lunched by an insider, like sabotage and data evasion, and also other attacks that SDN is vulnerable too, as discussed in Section 5.1. This approach is still being implemented by using ISCX2012 and CIDIS2017 intrusion detection datasets to first define a normal behavior and then to perform anomaly detection on the malicious packets stored in those datasets, both by using Random Forest, J48 and Naive Bayes algorithms. Therefore, we are also defining and implementing specific counters for our SDN switches by using P4 on a way that traffic classification can be not flow based only but also based on packet information, so that the encrypted protocol can be more effectively identified.

Our main goal in this research was to identify how encrypted attacks could be identified and stopped. We found that encrypted data can be analyzed by using metadata and develop an analysis strategy based on protocol analysis, modification and statistics. Our experiments, presented in Chapter 6 show that our methods were effective in detecting port scan and DoS in SDN, even when the flows are encrypted, being being non-intrusive and lightweight regarding to network bandwidth overhead, CPU and memory usage. Our results, presented on the experimental evaluation (see Chapter 6, show the effectiveness of these methods in detecting and blocking malicious flows through statistics. We also showed that our system is lightweight when considering resource consumption, such as network bandwidth switching, memory and processing power usage. Moreover, low false positive and negative rates were observed in our simulations. Although we are using only two main metadata or features provided by SDN (packets-per-second and bytes-per-second), others could be used (See Section 2.1) to improve detection and extend the proposed methods to protect SDN against other attacks, as the discussed in Section 2.3 and Section 5.1.

It is important to notice that, in the literature, there are few works related to detecting port scan and encrypted DoS for SDN. In addition, the available solutions do not have protection against them, which makes this work a relevant and promising alternative for detecting and preventing malicious threats. Besides, as our system works on the SDN controller, it may exchange information with other controllers, allowing a collaborative intrusion detection and prevention system, providing more security on SDN infrastructures.

7.1 Limitations and future work

Security still plays a fundamental role in computing environments. New network and computing technologies, such as SDN, are emerging and security must be improved to cope with these new technologies. The rise number of connected devices with Machine-to-Machine communications, specially in industrial environments, also increases security needs and challenges. Thus, we present some limitations and future directions for our research.

As our experiments are based on an experimental environment, there might be some actual existing problems that arise in a real networks environment and traffic that we did not tackled. Therefore, a real SDN environment and even a hybrid network (SDN and IP-based networks) should be used to evaluate accuracy and performance of our methods. Moreover, other overheads, such as the IPS database memory and CPU usage must be evaluated individually in order to improve even more the performance of our IPS, as we only consider the resources usage by the IPS itself in this work.

The proposed protection mechanism uses permanent rules on the SDN switches, which is typically a hardware with low hardware resources. Thus, our IPS could overload the switch flow table (memory capacity) with a large number of protection rules and so creating a new problem. Therefore, this should be a next step in improving our IPS. Although OpenFlow is a the *de facto* protocol for implementing SDN southbound communication, nowadays there are other proposals (see Section 2.1). Thus, it is important to evaluate such protocols in encrypted attacks detection. OpenFlow also provides specific counters data that could be extended by using P4. Thus, new counters could be defined to improve detection and specially encrypted traffic classification, which could be made based on packet information and not only in flow information. The communication and store of data that is encrypted by using homomorphic encryption schemes should also be investigated. Finally, the possible adoption of homomorphic encryption schemes in several environments could also offer new research possibilities to improve malicious traffic detection in encrypted data, as the traffic can be evaluated and analyzed without decryption.

7.2 Publications

During the development of this thesis we presented and discussed the results of our research in the following papers:

- NEU, C. V.; TATSCH, C.; LUNARDI, R.; MICHELIN, R.; OROZCO, A.; ZORZO, A. F. “Lightweight IPS for Port Scan in OpenFlow SDN Networks”. In: Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS), 2018, pp.

1–12. DOI: 10.1109/NOMS.2018.8406313. (Best paper of IEEE/IFIP Workshop on Security for Emerging Distributed Network Technologies).

- NEU, C. V.; ZORZO, A. F.; OROZCO, A. M. S.; MICHELIN, R. A. “An Approach for Detecting Encrypted Insider Attacks on OpenFlow SDN Networks”. In: 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), 2016, pp. 210–215.
- OROZCO, A. M. S.; NEU, C. V.; MICHELIN, R. A.; ZORZO, A. F. “Security Analysis of Forwarding Strategies in Network Time Measurements Using OpenFlow”. In: 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), 2016, pp. 148–153
- FAGUNDES, B. J. ; NEU, C. V. ; ZORZO, A. F. ; OROZCO, A. M. S.; MICHELIN, R. A. "SNORTIK - Uma Integração do IDS SNORT e o Sistema de Firewall do MIKROTIK ROUTEROS". In: 14 Escola Regional de Redes de Computadores (ERRC), 2016, Porto Alegre/RS/BRAZIL.
- GARCIA, T.O. ; NEU, C. V. "Definição de Novas Regras para o IDS Snort em Redes Definidas por Software". In: 15 Escola Regional de Redes de Computadores (ERRC), 2017, Santa Maria/RS/BRAZIL.

During the thesis development, the following contributions, to other thesis, were also produced:

- MICHELIN, R. A.; ZORZO, A. F.; CAMPOS, M.B.; NEU, C. V.; OROZCO, A. M. S.; “Smartphone as a biometric service for web authentication”. In: Proceedings of 11th International Conference for Internet Technology and Secured Transactions (ICITST), 2016.
- LUNARDI, R.; MICHELIN, R.; NEU, C. V.; ZORZO, A. “Distributed access control on IoT ledger-based architecture”. In: Proceedings of IEEE/IFIP Network Operations and Management Symposium (NOMS), 2018. DOI: 10.1109/NOMS.2018.8406154.

Other papers are also being produced and should be submitted shortly. Paper "ZONFlow: Security and Performance Balancing in SDN Control Plane" is being concluded to submit to a scientific journal. Paper " Detecting encrypted attacks in OPCUA" is also being concluded and will be submitted to a scientific conference. Other papers are being planned and will also be submitted to conferences: a)Detecting encrypted DoS in SDN; b) Encrypted traffic classification in SDN; and c) SIEM logs management based on ITIL. Those papers are also part of the research developed in this thesis.

Another work of our research group entitled "A Lightweight Blockchain for Industrial IoT" was submitted to the IEEE Transactions on Industrial Informatics. This work is currently under review.

REFERENCES

[AAG+15] Akhunzada, A.; Ahmed, E.; Gani, A.; Khan, M. K.; Imran, M.; Guizani, S. “Securing software defined networks: taxonomy, requirements, and open issues”, *IEEE Communications Magazine*, vol. 53–4, April 2015, pp. 36–44.

[AHR+10] Ali, S.; Haq, I. U.; Rizvi, S.; Rasheed, N.; Sarfraz, U.; Khayam, S. A.; Mirza, F. “On mitigating sampling-induced accuracy loss in traffic anomaly detection systems”, *ACM SIGCOMM Computer Communication Review*, vol. 40–3, July 2010, pp. 4–16.

[AM17] Anderson, B.; McGrew, D. “Machine learning for encrypted malware traffic classification: Accounting for noisy labels and non-stationarity”. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1723–1732.

[AMG07] Auld, T.; Moore, A. W.; Gull, S. F. “Bayesian neural networks for internet traffic classification”, *IEEE Transactions on Neural Networks*, vol. 18, January 2007, pp. 223 – 239.

[AN12] Alcock, S.; Nelson, R. “Libprotoident: traffic classification using lightweight packet inspection”, *WAND Network Research Group, Technical report*, 2012.

[APM17] Anderson, B.; Paul, S.; McGrew, D. “Deciphering malware’s use of tls (without decryption)”, *Journal of Computer Virology and Hacking Techniques*, vol. 14, August 2017, pp. 195–211.

[BAP+16] Bull, P.; Austin, R.; Popov, E.; Sharma, M.; Watson, R. “Flow based security for iot devices using an sdn gateway”. In: Proceedings of the IEEE 4th International Conference on Future Internet of Things and Cloud, 2016, pp. 157–163.

[BDG+14] Bosshart, P.; Daly, D.; Gibb, G.; Izzard, M.; McKeown, N.; Rexford, J.; Schlesinger, C.; Talayco, D.; Vahdat, A.; Varghese, G.; Walker, D. “P4: Programming protocol-independent packet processors”, *ACM SIGCOMM Computer Communication*, vol. 44, July 2014, pp. 87–95.

[BNR+17] Boite, J.; Nardin, P. A.; Rebecchi, F.; Bouet, M.; Conan, V. “Statesec: Stateful monitoring for ddos protection in software defined networks”. In: Proceedings of the IEEE Conference on Network Softwarization, 2017, pp. 1–9.

[Bra17] Braadland, A. S. “Key management for data plane encryption in sdn using wireguard”, *Master thesis, NTNU, Institutt for informasjonssikkerhet og kommunikasjonsteknologi, Oslo*, 2017.

[Bro] Bro. “Bro network security monitor”. Source: https://www.bro.org, Last access: 03 october 2018.

[BZ17] Bertoglio, D. D.; Zorzo, A. F. “Overview and open issues on penetration test”, *Journal of the Brazilian Computer Society*, vol. 23, February 2017, pp. 1–16.

[BsC+16] Beunardeau, M.; École normale supérieure; Connolly, A.; Geraud, R.; Naccache, D. “Fully homomorphic encryption: Computations with a blindfold”, *IEEE Security Privacy*, vol. 14, March 2016, pp. 63–67.

[CB15] Christodoulou, V.; Bi, Y. “A combination of cusum-ewma for anomaly detection in time series data”. In: Proceedings of the IEEE International Conference on Data Science and Advanced Analytics, 2015, pp. 1–5.

[CCAC11] Condon, E.; Cummins, E.; Afoulki, Z.; Cukier, M. “How secure are networked office devices?” In: Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems Networks, 2011, pp. 526–540.

[CEBBR14] Carela-Español, V.; Bujlow, T.; Barlet-Ros, P. “Is our ground-truth for traffic classification reliable?” In: Proceedings of the International Conference on Passive and Active Network Measurement, 2014, pp. 98 – 108.

[CER18] CERT.br. “Centro de estudos, resposta e tratamento de incidentes de segurança no brasil”. Source: http://www.cert.br/, Last access: 03 may 2018.

[Cisa] Cisco. “Cisco application centric infrastructure”. Source: http://www.cisco.com/c/en/us/solutions/data-center-virtualization/ application-centric-infrastructure/index.html, Last access: 24 october 2016.

[Cisb] Cisco. “Cisco open sdn controller”. Source: http://www.cisco.com/c/enlus/ products/cloud-systems-management/open-sdn-controller/index.html, Last access: 24 october 2016.

[Cisc] Cisco. “Encrypted traffic analytics”. In: White Paper. Source: https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/enterprise-network-security, Last access: 15 august 2018.

[Cis15] Cisco. “Openflow performance testing white paper”. Source: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-731302.pdf, Last access: 20 november 2015.

[CLLL16] Chen, C.; Li, B.; Lin, D.; Li, B. “Software-defined inter-domain routing revisited”. In: Proceedings of the IEEE International Conference on Communications, 2016, pp. 1–6.

[Clo] CloudStack, A. “Apache cloudstack: Open source cloud computing”. Source: http://cloudstack.apache.org, Last access: 03 september 2018.

[CMT12] Cappelli, D. M.; Moore, A. P.; Trzeciak, R. F. “The CERT guide to insider threats: how to prevent, detect, and respond to information technology crimes (Theft, Sabotage, Fraud)”. Addison-Wesley Professional, 1 edition, 432 pages, 2012.

[Com18] Community, S. “The snort project”. Source: https://snort.org, Last access: 03 october 2018.

[DAR] DARPA. “Kdd cup 1999 data”. Source: http://www.kdd.ics.uci.edu/databases/kddcup99/task.html, Last access: 03 september 2016.

[DDZX16] Dong, P.; Du, X.; Zhang, H.; Xu, T. “A detection method for a novel ddos attack against sdn controllers by vast new low-traffic flows”. In: Proceedings of the IEEE International Conference on Communications, 2016, pp. 1–6.

[DKMB15] Dauer, P.; Khondoker, R.; Marx, R.; Bayarou, K. “Security analysis of software defined networking applications for monitoring and measurement: Sflow and bigtap”. In: Proceedings of the International Conference on Future Internet, 2015, pp. 51–56.

[DMKK10] Dehghani, F.; Movahhedinia, N.; Khayyambashi, M. R.; Kianian, S. “Real-time traffic classification based on statistical and payload content features”. In: Proceedings of the International Workshop on Intelligent Systems and Applications, 2010, pp. 1–4.

[DRK+15] Detken, K. O.; Rix, T.; Kleiner, C.; Hellmann, B.; Renners, L. “Siem approach for a higher level of it security in enterprise networks”. In: Proceedings of the IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2015, pp. 322–327.

[Dro18] Dropbox, C. “Dropbox”. Source: https://www.dropbox.com, Last access: 10 october 2018.

[dSJAF+15] da Silva, A. S.; J.A.Wickbold; A.Schaeffer-Filho; A.K.Marnerides; A.Mauthe. “Tool support for the evaluation of anomaly traffic classification for network resilience”. In: Proceedings of the 20th IEEE Symposium on Computers and Communication, 2015, pp. 514–519.

[dSJAFG16] da Silva, A. S.; J.A.Wickbold; A.Schaeffer-Filho; Granville, L. Z. “Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn”. In: Proceedings of the IEEE/IFIP Network Operations and Management Symposium, 2016, pp. 27–35.

[dSMB+15] da Silva, A. S.; Machado, C. C.; Bisol, R.; Granville, L. Z.; A.Schaeffer-Filho. “Identification and selection of flow features for accurate traffic classification in sdn”. In: Proceedings of the IEEE 14th International Symposium on Network Computing and Applications, 2015, pp. 134–141.

[EGdSSGSF16] E. G. da Silva, A. S. da Silva, J. A. W.; Smith, P.; Granville, L. Z.; Schaeffer-Filho, A. “A one-class nids for sdn-based scada systems”. In: Proceedings of the IEEE 40th Annual Computer Software and Applications Conference, 2016, pp. 303–312.

[FNO+16] Fagundes, B.; Neu, C. V.; Orozco, A. M. S.; Michelin, R. A.; Zorzo, A. F. “Snortik - uma integração do IDS SNORT e o sistema de firewall do MikroTik RouterOS”. In: Anais da 14a Escola Regional de Redes de Computadores, 2016.

[Foua] Foundation, O. N. “Openflow switch specification 1.4.0”. Source: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.4.0.pdf, Last access: 07 mar 2018.

[Foub] Foundation, O. N. “Openflow switch specification, version 1.0.0”. Source: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf, Last access: 10 january 2018.

[Fouc] Foundation, O. N. “Openflow switch specification, version 1.3.0”. Source: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf, Last access: 08 october 2018.

[Foud] Foundation, O. N. “Threat analysis for the sdn architecture”. Source: https://www.opennetworking.org/wp-content/uploads/2014/10/Threat_Analysis_for_the_SDN_Architecture.pdf, Last access: 05 october 2018.

[Foue] Foundation, O. N. “Threat analysis for the SDN architecture”. Source: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/Threat_Analysis_for_the_SDN_Architecture.pdf, Last access: 10 december 2018.

[Fouf] Foundation, O. U. A. “Opc unified architecture pioneer of the 4th industrial (r)evolution”. Source: https://opcfoundation.org/wp-content/uploads/2014/03/OPC_UA_I_4.0_Pioneer_US_v2.pdf, Last access: 03 august 2018.

[Fou11] Foundation, O. N. “Open networking foundation website”. Source: https://www.opennetworking.org, Last access: 03 november 2018.

[Fou14] Foundation, O. N. “Openflow switch specification 1.5.0.” Source: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf, Last access: 10 october 2016.

[FS03] Ferguson, N.; & Schneier, B. “Practical Cryptography”. New York: John Wiley & Sons, 2003.

[GAA+14] Giotis, K.; Argyropoulos, C.; Androulidakis, G.; Kalogeras, D.; Maglaris, V. “Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments”, *Computer Networks*, vol. 62, 2014, pp. 122 – 136.

[Gar] Gartner. “Are cybercriminals hiding in your ssl traffic?” Source: https://whitepapers.theregister.co.uk/paper/view/3670/are-cybercriminals-hiding-in-your-ssl-traffic., Last access: 13 november 2018.

[GBC16] Goransson, P.; Black, C.; Culver, T. “Software Defined Networks, Second Edition: A Comprehensive Approach”. Morgan Kaufmann, 2016, 2 ed..

[GJW+14] Go, Y.; Jeong, E.; Won, J.; Kim, Y.; Kune, D. F.; Park, K. “Gaining control of cellular traffic accounting by spurious tcp retransmission”. In: Network and Distributed System Security Symposium, 2014, pp. 1–15.

[Groa] Group, M. W. “Mawi working group traffic archive”. Source: http://mawi.wide.ad.jp/mawi, Last access: 19 september 2018.

[Grob] Group, N. W. “Rfc4741: Netconf configuration protocol”. Source: https://tools.ietf.org/html/rfc4741, Last access: 03 december 2018.

[GZL06] Gao, M.; Zhang, K.; Lu, J. “Efficient packet matching for gigabit network intrusion detection using tcams”. In: Proceedings of the 20th International

Conference on Advanced Information Networking and Applications, 2006, pp. 249–254.

[HHB14] Hu, F.; Hao, Q.; Bao, K. “A survey on software-defined network and openflow: From concept to implementation”, *Communications Surveys Tutorials, IEEE*, vol. 16, Fourthquarter 2014, pp. 2181–2206.

[HPSR18] Hofstede, R.; Pras, A.; Sperotto, A.; Rodosek, G. D. “Flow-based compromise detection: Lessons learned”, *IEEE Security Privacy*, vol. 16, February 2018, pp. 82–89.

[HSDK15] Haleplidis, E.; Salim, J.; Denazis, S.; Koufopavlou, O. “Towards a network abstraction model for sdn”, *Journal of Network and Systems Management*, vol. 23, 2015, pp. 309–327.

[IET] IETF, I. E. T. F. “Opflex control protocol”. Source: https://tools.ietf.org/html/draft-smith-opflex-03, Last access: 15 november 2018.

[IET11] IETF, I. E. T. F. “Rfc6241: Network configuration protocol (netconf)”. Source: https://tools.ietf.org/html/rfc4741, Last access: 11 december 2018.

[IET13] IETF, I. E. T. F. “Rfc7047: The open vswitch database management protocol”. Source: https://tools.ietf.org/html/rfc7047, Last access: 03 october 2018.

[IET16] IETF, I. E. T. F. “An architecture for the interface to the routing system (rfc 7921)”. Source: https://tools.ietf.org/html/rfc7921, Last access: 19 october 2018.

[JGSG17] Jazi, H. H.; Gonzalez, H.; Stakhanova, N.; Ghorbani, A. “Detecting http-based application layer dos attacks on web servers in the presence of sampling”, *Computer Networks*, vol. 121, July 2017, pp. 25–36.

[KF13] Kim, H.; Feamster, N. “Improving network management with software defined networking”, *IEEE Communications Magazine*, vol. 51–2, February 2013, pp. 114–119.

[KGR14] Koch, R.; Golling, M.; Rodosek, G. “Behavior-based intrusion detection in encrypted environments”, *IEEE Communications Magazine*, vol. 52, 2014, pp. 124–131.

[KID17] Kolpyakwar, A. A.; Ingle, M. G.; Deshmukh, R. V. “A survey on data mining approaches for network intrusion detection system”, *International Journal of Computer Applications*, vol. 159, 2017, pp. 20–23.

[KKB07] Koh, K.; Kim, S.; Boyd, S. “An interior-point method for large-scale l1-regularized logistic regression”, *Journal of Machine learning research*, vol. 8, January 2007, pp. 1519–1555.

[Koc09] Koch, R. “Changing network behavior”. In: Proceedings of the IEEE Third International Conference on Network and System Security, 2009, pp. 60–66.

[KR16] Kumar, M. G. V.; Ragupathy, U. S. “A survey on current key issues and status in cryptography”. In: Proceedings of the International Conference on Wireless Communications, Signal Processing and Networking, 2016, pp. 205–210.

[KREV+15] Kreutz, D.; Ramos, F.; E. Verissimo, P.; E. Rothenberg, C.; Azodolmolky, S.; Uhlig, S. “Software-defined networking: A comprehensive survey”, *Proceedings of the IEEE*, vol. 103, January 2015, pp. 14–76.

[KRV13] Kreutz, D.; Ramos, F. M. V.; Verissimo, P. “Towards secure and dependable software-defined networks”. In: Proceendings of the second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, 2013, pp. 55–60.

[KXF18] Knob, L. A. D.; Xavier, B.; Ferreto, T. “An unikernels provisioning architecture for openstack”. In: Proceedings of the IEEE Symposium on Computers and Communications, 2018, pp. 903–908.

[KZMB14] Khondoker, R.; Zaalouk, A.; Marx, R.; Bayarou, K. “Feature-based comparison and selection of software defined networking (sdn) controllers”. In: Proceedings of the Computer Applications and Information Systems, 2014, pp. 1–7.

[LCCY17] Liu, Y.; Chen, J.; Chang, P.; Yun, X. “A novel algorithm for encrypted traffic classification based on sliding window of flow’s first n packets”. In: Proceeddings of the 2ND IEEE International Conference on Computer Intelligence and Applications, 2017, pp. 463–470.

[lCFCG18] ling Chan, C.; Fontugne, R.; Cho, K.; Goto, S. “Monitoring tls adoption using backbone and edge traffic”. In: Proceedings of the IEEE Conference on Computer Communications Workshops, 2018, pp. 208 – 213.

[LF17] Lagraa, S.; François, J. “Knowledge discovery of port scans from darknet”. In: Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management, 2017, pp. 935–940.

[LHA+18] Li, S.; Han, K.; Ansari, N.; Bao, Q.; Hu, D.; Liu, J.; Yu, S.; Ahu, Z. “Improving sdn scalability with protocol-oblivious source routing: A system-level study”, *IEEE Transactions on Network and Service Management*, vol. 15, March 2018, pp. 275–288.

[LHM10] Lantz, B.; Heller, B.; McKeown, N. “A network in a laptop: Rapid prototyping for software-defined networks”. In: Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, pp. 1–6.

[Ltd18] Ltd, C. “Ubuntu”. Source: https://www.ubuntu.com, Last access: 30 october 2018.

[Lyo09] Lyon, G. F. “Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning”. USA: Insecure, 2009.

[Lyo18] Lyon, G. “Nmap security scaner”. Source: https://insecure.org, Last access: 30 october 2018.

[MAB+08] McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. “Openflow: Enabling innovation in campus networks”, *ACM SIGCOMM Computer Communication Review*, vol. 38, March 2008, pp. 69–74.

[McC18] McCauley, M. “Pox”. Source: http://www.noxrepo.org, Last access: 30 september 2018.

[MdAN+11] Mozzaquatro, B. A.; de Azevedo, R. P.; Nunes, R. C.; Cappo, C.; de Jesus Kozakevicius, C. S. A. “Anomaly-based techniques for web attacks detection”, *Journal of Applied Computing Research*, vol. 1, December 2011, pp. 111–120.

[MFH18] Mahdavi, E.; Fanian, A.; Hassannejad, H. “Encrypted traffic classification using statistical features”, *The ISC International Journal of Information Security*, vol. 10, January 2018, pp. 29–43.

[Min18] Mininet. “Mininet simulator: An instant virtual network on your laptop”. Source: http://mininet.org, Last access: 03 may 2018.

[MMIoT] MIT Massachusetts Institute of Technology, L. L. “Darpa, defense advanced research projects agency for intrusion detection evaluation”. Source: https://www.ll.mit.edu/ideval/docs/attackDB.html, Last access: 03 october 2015.

[MRFFFRAB15] Martínez-Rego, D.; Fernández-Francos, D.; Fontenla-Romero, O.; Alonso-Betanzos, A. “Stream change detection via passive-aggressive classification and bernoulli cusum", *Information Sciences*, vol. 305, June 2015, pp. 130–145.

[MSSK18] McGaughey, D.; Semeniuk, T.; Smith, R.; Knight, S. "A systematic approach of feature selection for encrypted network traffic classification". In: Proceedings of the Annual IEEE International Systems Conference, 2018, pp. 463–470.

[MVTG14] Medved, J.; Varga, R.; Tkacik, A.; Gray, K. "Opendaylight: Towards a model-driven sdn controller architecture". In: IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2014, pp. 1–6.

[MZR14] Michelin, R. A.; Zorzo, A. F.; Rose, C. A. D. "Mitigating dos to authenticated cloud rest apis". In: Proceedings of the 9th International Conference for Internet Technology and Secured Transactions, 2014, pp. 106–111.

[NPSR16] Neelam, D.; Prasenjit, M.; Shashank, S.; Rahamatullah, K. "Research trends in security and ddos in sdn", *Security and Communication Networks*, vol. 9, December 2016, pp. 6386–6411.

[NS14] Nishtha; Sood, M. "Software defined network - architectures". In: Proceedings of the International Conference on Parallel, Distributed and Grid Computing, 2014, pp. 451–456.

[NTL$^+$18] Neu, C. V.; Tatsch, C.; Lunardi, R.; Michelin, R. A.; Orozco, A. M. S.; Zorzo, A. F. "Lightweight ips for port scan in openflow sdn networks". In: Proceedings of the IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1–6.

[NZD$^+$16] Nanda, S.; Zafari, F.; DeCusatis, C.; Wedaa, E.; Yang, B. "Predicting network attack patterns in sdn using machine learning approach". In: Proceedings of the IEEE Conference on Network Function Virtualization and Software Defined Networks, 2016, pp. 167–172.

[NZOM16] Neu, C. V.; Zorzo, A. F.; Orozco, A. M. S.; Michelin, R. A. "An approach for detecting encrypted insider attacks on openflow sdn networks", *Proceedings of the 11th International Conference for Internet Technology and Secured Transactions*, December 2016, pp. 210–215.

[OCD16] Osanaiye, O.; Choo, K.-K. R.; Dlodlo, M. "Change-point cloud ddos detection using packet inter-arrival time". In: Proceedings of the 8th Computer Science and Electronic Engineering, 2016.

[OML10] Owezarski, P.; Mazel, J.; Labit, Y. “0day anomaly detection made possible thanks to machine learning”. In: Proceedings of the 8th International Conference on Wired/Wireless Internet Communications, 2010, pp. 327–338.

[ONMZ16] Orozco, A. M. S.; Neu, C. V.; Michelin, R. A.; Zorzo, A. F. “Security analysis of forwarding strategies in network time measurements using openflow”. In: Proceedings of the 11th International Conference for Internet Technology and Secured Transactions, 2016, pp. 148–153.

[Ope] Open Networking Foundation. “Software-Defined Networking: The New Norm for Networks”. Source: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf, Last access: 09 september2018.

[Ope18a] Open Networking Foundation. “Openflow”. Source: https://opennetworking.org/sdn-resources/openflow, Last access: 02 may 2018.

[OPE18b] OPENDAYLIGHT. “Opendaylight, a linux foundation collaborative project”. Source: http://www.opendaylight.org, Last access: 30 september 2018.

[Ope18c] OpenStack. “Openstack open source cloud computing software”. Source: https://www.openstack.org, Last access: 03 september 2018.

[Ora18] Oracle. “Virtualbox”. Source: https://www.virtualbox.org, Last access: 30 october 2018.

[P418] P4, C. “P4 language and related specifications”. Source: https://p4.org, Last access: 30 september 2018.

[PAGT07] Peddabachigaria, S.; Abrahamb, A.; Grosanc, C.; Thomas, J. “Modeling intrusion detection system using hybrid intelligent systems”, *Journal of Network and Computer Applications*, vol. 30, January 2007, pp. 114–132.

[Pal15] Paladi, N. “Towards secure sdn policy management”. In: Proceedings of the IEEE/ACM 8th International Conference on Utility and Cloud Computing, 2015, pp. 607–611.

[Pax98] Paxson, V. “Bro: a system for detecting network intruders in real-time. computer networks”. In: Proceedings of the 7th USENIX Security Symposium, 1998, pp. 26–29.

[Pin17] Pinder, J. P. “Introduction to Business Analytics using Simulation”. Elsevier Inc., 2017, 1 ed..

[PPK+09] Pfaff, B.; Pettit, J.; Koponen, T.; Amidon, K.; Casado, M.; Shenker, S. “Extending networking into the virtualization layer”. In: Proceedings of the Workshop on Hot Topics in Networks, 2009, pp. 1–6.

[PR06] Peterson, L.; Roscoe, T. “The design principles of planetlab”, *ACM SIGOPS Operating Systems Review*, vol. 40, January 2006, pp. 11–16.

[Pre] Presuhn, R. “Rfc3436: Version 2 of the protocol operations for the simple network management protocol (snmp)”.

[Pro18] Project Floodlight. “Project floodlight: Open source software for building software-defined networks”. Source: http://www.projectfloodlight.org, Last access: 30 september 2018.

[PTL15] Parulkar, G.; Tofigh, T.; Leenheer, M. D. “Sdn control of packet-over-optical networks”. In: Proceedings of the Optical Fiber Communication Conference, 2015, pp. W1G.4.

[RNME+15] Rodriguez-Natal, A.; Marc, P.-C.; Ermagan, V.; Lewis, D.; Farinacci, D.; Maino, F.; Cabellos-Aparicio, A. “Lisp: a southbound sdn protocol?”, *IEEE Communications Magazine*, vol. 53, July 2015, pp. 201–207.

[Roe99] Roesch, M. “Snort - lightweight intrusion detection for networks”. In: Proceedings of the 13th USENIX Conference on System Administration, 1999, pp. 229–238.

[RV17] Rangasami, K.; Vagdevi, S. “Comparative study of homomorphic encryption methods for secured data operations in cloud computing”. In: Proceedings of the International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques, 2017.

[RYU18] RYU SDN FRAMEWORK COMMUNITY. “Ryu network operation system”. Source: https://osrg.github.io/ryu/, Last access: 30 september 2018.

[SAS+17] Stancu, A.; Avram, A.; Skorupski, M.; Vulpe, A.; Halunga, S. “Enabling sdn application development using a netconf mediator layer simulator”. In: Proceedings of the Ninth International Conference on Ubiquitous and Future Networks, 2017, pp. 658–663.

[SCM09] Sandouka, H.; Cullen, A. J.; Mann, I. “Social engineering detection using neural networks”. In: Proceedings of the International Conference on CyberWorlds, 2009, pp. 273–278.

[SD16] Sanzgiri, A.; Dasgupta, D. “Classification of insider threat detection techniques”. In: Proceedings of the 11th Annual Cyber and Information Security Research Conference, 2016, pp. 1–4.

[SDYL+] Smith, M.; Dvorkin, M.; Y. Laribi, V. P.; Garg, P.; Weidenbacher, N. “Opflex control protocol”. Source: https://tools.ietf.org/html/draft-smith-opflex-03, Last access: 19 november 2018.

[SEKC16] Salman, O.; Elhajj, I.; Kayssi, A.; Chehab, A. “Sdn controllers: A comparative study”. In: Proceedings of the 18th Mediterranean Electrotechnical Conference, 2016, pp. 1–6.

[SFF+13] Shah, S. A.; Faiz, J.; Farooq, M.; Shafi, A.; Mehdi, S. A. “An architectural evaluation of sdn controllers”. In: Proceedings of the IEEE International Conference on Communications, 2013, pp. 3504–3508.

[Shu13] Shukla, V. “Introduction to Software Defined Networking - OpenFlow & VxLAN”. CreateSpace Independent Publishing Platform, 2013, 1st ed..

[SKS15] Saxena, A.; Kaulgud, V.; Sharma, V. “Flow-application layer encryption for cloud”. In: Proceedings of the Asia-Pacific Software Engineering Conference, 2015, pp. 1–6.

[SLG18] Sharafaldin, I.; Lashkari, A. H.; Ghorbani, A. A. “Toward generating a new intrusion detection dataset and intrusion traffic characterization”. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy, 2018, pp. 108–116.

[SSR15] Seeber, S.; Stiemert, L.; Rodosek, G. D. “Towards an sdn-enabled ids environment”. In: IEEE Conference on Communications and Network Security, 2015, pp. 751–752.

[SSS+10] Sperotto, A.; Schaffrath, G.; Sadre, R.; Morariu, C.; Pras, A.; Stiller, B. “An overview of ip flow-based intrusion detection”, *IEEE Communications Surveys Tutorials*, vol. 12, April 2010, pp. 343–356.

[SSTG12] Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A. A. “Toward developing a systematic approach to generate benchmark datasets for intrusion detection”, *Computers Security*, vol. 31, May 2012, pp. 357–374.

[Sur10] Suricata. “Suricata open source ids/ips/nsm engine”. Source: https://suricata-ids.org,2010, Last access: 03 october 2018.

[SWJ15] Scandariato, R.; Wuyts, K.; Joosen, W. “A descriptive study of microsoft’s threat modeling technique”, *Requirements Engineering*, vol. 20, June 2015, pp. 163–180.

[Tü17] Türpe, S. “The trouble with security requirements”. In: Proceedings of the IEEE 25th International Requirements Engineering Conference, 2017, pp. 122 – 133.

[UNBa] UNB. “A network traffic biflow generator and analyzer (formerly iscxflowmeter)”. Source: http://netflowmeter.ca, Last access: 09 november 2018.

[UNBb] UNB-datasets. “Unb datasets”. Source: https://www.unb.ca/cic/datasets/index.html, Last access: 02 november 2018.

[UNBc] UNB-ISCX. “Unb iscx. intrusion detection evaluation dataset”. Source: http://www.unb.ca/research/iscx/dataset/iscx-IDS-datasep.html, 30 november 2018.

[Vac17] Vacca, J. R. “Computer and Information Security Handbook 3rd Edition”. Elsevier, 2017.

[VMC08] Viega, J.; Messier, M.; Chandra, P. “Network Security with OpenSSL: Cryptography for Secure Communications”. O’Reilly Media, 2008.

[VD15] Velan, P.; Čermák, M.; Čeleda, P.; Drašar, M. “A survey of methods for encrypted traffic classification and analysis”, *International Journal of Network Management*, vol. 25, July 2015, pp. 355–374.

[WCCL18] Wang, T.; Chen, H.; Cheng, G.; Lu, Y. “Sdnmanager: A safeguard architecture for sdn dos attacks based on bandwidth prediction”, *Security and Communication Networks*, vol. 2018, January 2018, pp. 52–59.

[WL06] Walton, R.; Limited, W.-M. “Balancing the insider and outsider threat”, *Computer Fraud & Security*, vol. 2006, November 2006, pp. 8–11.

[XC18] Xu, H.; Chen, H. “Constructing a basic-element library from snmp mibs for management information tree of software-defined networking”. In: Proceedings of the 37th Chinese Control Conference, 2018, pp. 6132 – 6136.

[XL09] Xu, P.; Lin, S. “Internet traffic classification using c4.5 decision tree”, *Journal of software*, vol. 20, January 2009, pp. 2692–2704.

[XTS$^+$12] Xu, D.; Tu, M.; Sanford, M.; Thomas, L.; Woodraska, D.; Xu, W. “Automated security test generation with formal threat models”, *IEEE Transactions on Dependable and Secure Computing*, vol. 9, July 2012, pp. 526–540.

[YWS$^+$14] Yu, Z.; Wang, X.; Song, J.; Zheng, Y.; Song, H. “Forwarding programming in protocol-oblivious instruction set”. In: Proceedings of the IEEE 22nd International Conference on Network Protocols, 2014, pp. 577 – 582.

[YY15] Yan, Q.; Yu, F. R. “Distributed denial of service attacks in software-defined networking with cloud computing”, *IEEE Communications Magazine*, vol. 53, April 2015, pp. 52–59.

[YZ08] Yuan, D.; Zhong, J. “A lab implementation of syn flood attack and defense”. In: Proceedings of the 9th ACM SIGITE Conference on Information Technology Education, 2008, pp. 57–58.

[ZGYW16] Zhang, L.; Guo, Y.; Yuwen, H.; Wang, Y. “A port hopping based DoS mitigation scheme in SDN network”. In: Proceedings of the 12th International Conference on Computational Intelligence and Security, 2016, pp. 314–317.

[ZHKS16] Zolotukhin, M.; Hämäläinen, T.; Kokkonen, T.; Siltanen, J. “Increasing web service availability by detecting application-layer ddos attacks in encrypted traffic”. In: Proceedings of the 23rd International Conference on Telecommunications, 2016, pp. 1–6.

[ZKF$^+$17] Zhang, Z.; Kang, C.; Fu, P.; Cao, Z.; Li, Z.; Xiong, G. “Metric learning with statistical features for network traffic classification”. In: Proceedings of the IEEE 36th International Performance Computing and Communications Conference, 2017, pp. 1 – 7.