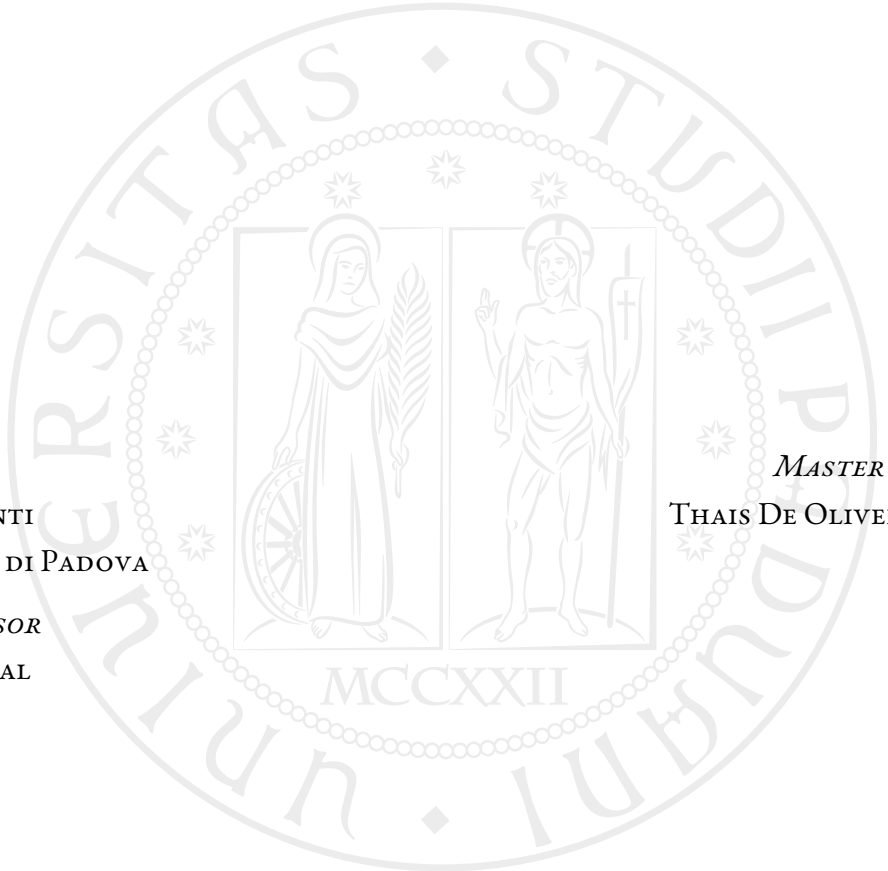
Università degli Studi di Padova

Department of Information Engineering

Master Thesis in ICT for Internet and Multimedia Engineering

Security attacks and solutions on SDN control plane: A survey



Supervisor
Mauro Conti
Università di Padova

Co-supervisor
Chhagan Lal
Tu Delft

Master Candidate
Thais De Oliveira Gomese

To the council of professors who admitted me to this university.

To my beloved Italian flatmate Stefano Ronchi for his help, advises and laughs.

To my sister Beatriz and my mother Telma for all care and support during this crazy end of the semester.

To my beloved relatives: aunt Rosenir, aunt Guiomar and aunt Nadir.

All my love to my father Robson, my grandfather Wilson,my grandmother Nair, uncle Nelson and his adorable father now seeing me from the stars."Sim, o Galileu deu aula na minha universidade,pai!"

Abstract

Software Defined Networks (SDN) is an open programmable network model promoted by ONF that has been a key-enabler of recent technology trends. SDN explores the separation of data and control plane. Different from the past concepts, SDN introduces the idea of separation of the control plane (routing and traffic decisions) and data plane (forwarding decisions based on the control plane) that challenges the vertical integration achieved by the traditional networks, in which network devices such as router and switches accumulate both functions.

SDN presents some advantages such as centralized management and the ability to be programmed on demand. Apart from these benefits, SDN still presents security vulnerabilities and among them, the most lethal ones are targeting the control plane. As the controllers residing on the control plane manages the underlying networking infrastructure and devices (i.e., routers/switches), any security threat, malware, or issues during the carrying out of activities by the controller can lead to disruption of the entire network. In particular, due to its centralized position, the (SDN) controller is seen as a single point of failure. As a result, any attack or vulnerability targeting the control plane or controller is considered fatal to the point of disrupting the whole network. In this thesis, the security threats and attacks targeting the (SDN) control plane are identified and categorized into different groups by considering how they cause an impact to the control plane.

To obtain results, extensive literature research has been carried out by performing an in-depth study of the existing research articles that discusses an array of attacks and their corresponding solutions for the (SDN) control plane. Mainly, the solutions intended to detect, mitigate, or protect the (SDN) control plane against potential threats and attacks have been considered. On basis of this task, the potential articles selected were categorized with respect to their impact to the (SDN) control plane as direct and indirect. Where applicable a comparison of the solutions addressing the same attack has been provided. Moreover, the advantages and disadvantages of the solutions addressing the respective attacks are presented. Finally, a discussion regarding the findings and results obtained during this surveying process and future work suggestions extracted during the review process have been discussed.

Keywords: SDN, Security, Control Plane, Denial of Service, Topology Attacks, Openflow

Sommario

Software Defined Networks (SDN) è un modello di rete programmabile aperto promosso da ONF , che è stato un fattore chiave per le recenti tendenze tecnologiche. SDN esplora la separazione dei dati e del piano di controllo . Diversamente dai concetti passati, SDN introduce l'idea di separazione del piano di controllo (decisioni di instradamento e traffico) e piano dati (decisioni di inoltro basate sul piano di controllo) che sfida l'integrazione verticale raggiunta dalle reti tradizionali, in cui dispositivi di rete come router e switch accumulano entrambe le funzioni.

SDN presenta alcuni vantaggi come la gestione centralizzata e la possibilità di essere programmato su richiesta. Oltre a questi vantaggi, SDN presenta ancora vulnerabilità di sicurezza e, tra queste,le più letali prendono di mira il piano di controllo. Come i controllers che risiedono sul piano di controllo gestiscono l'infrastruttura e i dispositivi di rete sottostanti (es. router/switch), anche qualsiasi insicurezza, minacce, malware o problemi durante lo svolgimento delle attività da parte del controller, possono causare interruzioni dell'intera rete. In particolare, per la sua posizione centralizzata, il controller SDN è visto come un punto di fallimento. Di conseguenza, qualsiasi attacco o vulnerabilità che prende di mira il piano di controllo o il controller è considerato fatale al punto da sconvolgere l'intera rete. In questa tesi, le minacce alla sicurezza e gli attacchi mirati al piano di controllo (SDN) sono identificati e classificati in diversi gruppi in base a come causano l'impatto sul piano di controllo.

Per ottenere risultati, è stata condotta un'ampia ricerca bibliografica attraverso uno studio approfondito degli articoli di ricerca esistenti che discutono di una serie di attacchi e delle relative soluzioni per il piano di controllo SDN. Principalmente, come soluzioni intese a rilevare, mitigare o proteggere il (SDN) sono stati presi in considerazione le potenziali minacce gli attachi al piano di controllo. Sulla base di questo compito, gli articoli selezionati sono stati classificati rispetto al loro impatto potenziale sul piano di controllo (SDN) come diretti e indiretti. Ove applicabile, è stato fornito un confronto tra le soluzioni che affrontano lo stesso attacco. Inoltre, sono stati presentati i vantaggi e gli svantaggi delle soluzioni che affrontano diversi attacchi . Infine, una discussione sui risultati e sui esitti ottenuti durante questo processo di indagine e sono stati affrontatti suggerimenti di lavoro futuri estratti durante il processo di revisione.

Parole chiave : SDN, Sicurezza, Piano di controllo, Denial of Service, Attacchi alla topologia, Openflow

Contents

Listing of figures

Listing of tables

1 Introduction

Software Defined Networks (SDN) is a cutting-edge technology that is revolutionizing traditional network communication. It brings new concepts to facilitate the deployment of real-time decisions in a simple way. In contrast with the current technology that has been proved limited to adjust to the most recent trends in technology such as Big Data, Cloud Computing. SDN can deliver high quality services for a large number of users. The Opening Networking Foundation (ONF), a non-profit consortium in charge of standardization of SDN architecture, lists some of the limitations of the traditional networks [1]:

- **Complexity and static architecture**
  To attend to the claims of the modern interconnected world, research centers and companies have developed protocols to provide network services to users and industry. Considering the multiplicity of network services with its solutions, consequently, diverse protocols. Some proprietary protocols carry hardware and permission restrictions. Each one of them working independently and requiring certain hardware and network requirements. However, this isolation is broken when some activities are carried out. The example provided by Opening Networking Foundation (ONF) [1], the task of moving or adding a new device can impact the functioning of other devices such as routers ,switches, firewalls and updates to ACL, setting. This action can lead to topology, protocols, software, and hardware conflicts. Along with all this complexity to perform operations, the lack of dynamism of network architecture leads to arduous configuration processes for IT administrators. Simple tasks such as configuration turn into a manual process in which the IT expert must configure features of every single device respecting the vendor's requirements and restrictions.

- **Inconsistencies policies**
  The addition of a new security policy to an enterprise can impact thousands of devices. As mentioned previously, configuring different devices from diverse vendors under different requirements is a slow process. Furthermore, the endorsement of policies can take a long time to be propagated. In case of bugs, defects, or any human configuration errors during this propagation process can result in a lack of consistency between systems.

- **Inability to scale**
  Large companies are expanding their business and adapting to a complex technological scenario with fine-grained services and the inclusion of thousands of devices. To expand their business

and support the high volume of data and processing, it is necessary for hyper scale networks to deliver services with high quality and accommodate the number of users.

- **Vendor dependency**
  The industry's reaction to satisfy business needs and user demands is limited by the vendor's product cycle. As an example, a company might be interested in implementing a solution with certain vendor's equipment that later it is discovered that the most recent release is not compatible with other devices.

Considering all the problems faced, ONF has started the SDN movement to overcome all the limitations of the current technology paradigm and to attend to the market and business needs along with other standardization bodies and consortiums such as ITU, IEEE, IETF/IRFT,and 3GPP [2]. In terms of advantages, SDN brings a more up-to-date and innovative approach in comparison to the more traditional network such as [3]:

- **Agile**
  The network administrator or programmers can configure the network traffic according to the business needs or demands in run-time.

- **Centrally managed**
  The routing and topology decisions are centralized in the SDN controller that provides a global view of switches and hosts and enforce network policies.

- **Programmability**
  The network administrators can write programs, manage, update resources without being bound to a specific vendor, devices, and protocols, or proprietary software.

1.1 Motivations

This thesis aims to bring light to the most recent and more frequent vulnerabilities and attacks that target the SDN control plane. Multiple surveys have been launched over the past seven years covering different aspects and with diverse emphasis on the layers of the SDN stack. Even though countless comprehensive surveys and review articles have been already published so far, this thesis attempts to update a broad range of attacks with their corresponding available solutions.

Indeed, the idea is to facilitate the comprehension of the curious reader and enthusiast of network security. In some ways, the work tries to present a different perspective of all the past literature by focusing exclusively on the control plane, the main component and the most dreadful part of the SDN network. Any impact, or vulnerability can take down the whole network, degrade performance, and exposure details to be manipulated by attackers.

Although some mitigation and detection techniques have been presented over the last ten years, there is still a vast field to be explored by researchers. Additionally, even robust mechanisms present limitations, weaknesses, vulnerabilities, and disadvantages that can be explored for malignant purposes.

For this reason, the main objective of this work are to summarize and categorize attacks with the corresponding solutions to be implemented or to be used as a ground basis for continuous improvement and investigation. Ideally, presenting vulnerabilities of the solutions proposed could be a great opportunity to expose a problem to future investigation.

The categorization method is defined as the action of grouping ideas or objects that share any degree of similarity. Science research deals with a large and diffuse production of articles and papers that makes any process of grouping areas essential to facilitate the access and comprehension of the subject to all members of the research community.

The categorization method applied to Software Defined Networks aims to redirect the reader from diffuse solutions to narrow ones. Some categories presented similarities to others. However, the decisive point was the emphasis given to a particularity of the grouping of objects [4].

1.2 Contributions

The objective and contributions of this thesis are as follows :

- I present a summary of the most frequent and recently discovered security issues and attacks targeting the SDN control plane categorized in terms of direct and indirect impact. The investigated attacks were grouped per their constituent to the Northbound Interface and Southbound Interface for their easy understanding and presentation.
- Synthesize available literature solutions to secure the SDN control plane from different attacks and threats. Multiple surveys have been launched over the past seven years covering different aspects and with diverse emphasis on the layers of the SDN stack. However, there are no comprehensive SDN security surveys in recent years. This survey diverges from the others in respect to focus exclusively on the control plane which is the most critical place to position an attack and it includes a large number of recent control plane attacks an their corresponding solutions which are missing from the existing surveys.
- Based on my investigation of a large number of security attacks and their solutions related to the SDN control plane, a discussion of the possible solutions for some of the under researched attacks. Moreover, future works in security that could affect a SDN network preparing the path to future enhancements are presented to be used as a ground for consistent research.

1.3 Organization

In the next chapters, it will be explained in more detail the technical background that inclines the adherence of this technology, while this section was in respect to the driving forces and international efforts behind SDN. chapter 2 introduces the technical background, the definitions regarding SDN technologies and the related work that resulted in this thesis. chapter 3 presents the attacks and security issues surveyed and solutions. chapter 4 introduces the security solutions available in the literature. Finally, Chapter 5 finalizes with a discussion regarding the solutions found.

2

Background and Related Works

This chapter presents the basic concepts to contextualize the reader of the three basic plans of SDN: Application Plane, Control Plane, and Data Plane. Followed by the two interfaces that bridges the planes: Northbound Interface and Southbound Interface. In the Related Works section, it is shown the existing reviews that become the starting point for the creation and main objectives of this one.

2.1 Introduction to SDN

Software Defined Networks is an open programmable network model in which the network devices are servants to a central entity called a controller that supervises and deploys programs under the same Application Programming Interface (API). Different from the past concepts of the networks, SDN introduces the idea of separation of the control plane (routing and traffic decisions) and data plane (forwarding decisions based on the control plane) challenges the vertical integration achieved by the traditional networks, in which network devices such as routers and switches accumulate both functions.

The control plan consists of one or more of the entities called controllers which correspond to the "brain" of the system where all the routing decisions are made, while the data plane comprises intermediate nodes such as switches, which are reduced to only propagators of the outcomes from the upper layer as it is illustrated in the fig. 2.1. The controller responds to programming commands given by network administrators. Endorsement of security policies, response to threats, and network change status are sent from the application layers to the controllers to be translated into low-level commands to the data plane.

The data exchange between these plans is implemented by Application Programming Interface (API) to standardize the communication between the planes regardless of the vendor technology implemented.The most prominent is Open Flow, which is a protocol that has been guaranteeing the success of SDN. OpenFlow is largely adopted by network hardware vendors. This protocol facilitates developers to control packet forwarding.

SDN would have not been a reality without the development of OpenFlow. The first version 1.0.0 was released in December 2009, even though past releases for experimental purposes were launched

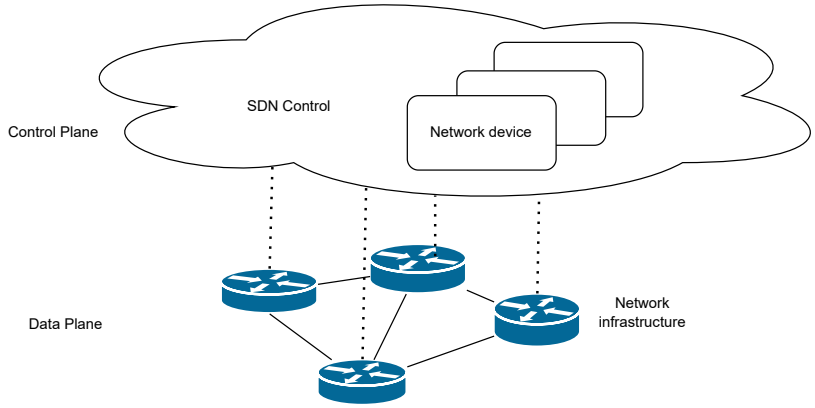


Figure 2.1: Data plane and control plane scheme.

Source: Stallings [5]

before that date. Until the conception of version 1.1.0, the management and development of specifications were under openflow.org until the creation of ONF in March 2011, an institution responsible for the democratization and advance of innovations in SDN. Since then ONF became in charge of the evolution of OpenFlow specifications starting from version 1.1 [6].

Open Networking Foundation which is an industry consortium that is responsible for the promotion and implementation through providing interoperability testing along with many other events in order to guarantee its use by diverse technology vendors. The facilitation of the integration with the current infrastructure technology has been proved simple due to a simplified firmware or software updates which resulted in a large acceptance of the market [1].

OpenFlow permits the real-time definition, alteration, and modification of the traffic according to the choice of the network administrator and facilitates the network security policies deployment, basically supports network management and allows convenient programming of the network devices.

In summary, SDN presents key particularities that differentiate from the previous concepts such as [2]:

- The centralization of decisions in a singles entity, in other words, the separation of the control and data plane
- The controller has a centralized and abstracted view of the network and can apply modifications, reinforce policies on run-time
- The utilization of APIs to enable the communication between the planes
- The on-demand programmability results in a more flexible approach to attending to business demands and responding to security threats.

2.1.1 SDN Architecture

The idea behind SDN can be easily understood by an analogy of an Operating System running on a network device as it is shown in fig. 2.2. From the left perspective, the device can be broken into controller cards and interface cards. The interface card is in charge of forwarding the packets according

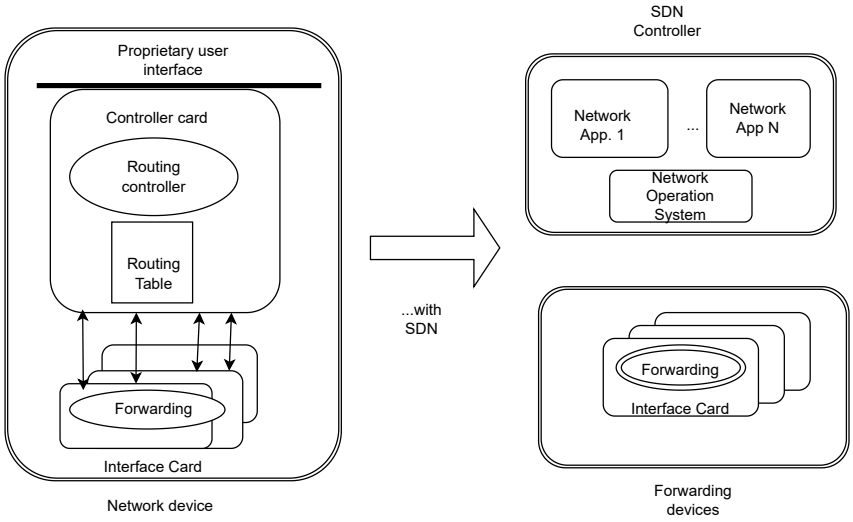


Figure 2.2: Comparison of SDN and network device.

Source: Costanzo et al. [3]

to the instruction given by the forwarding table (copy repository of the routing table containing the next interface hop or destination IP). The routing table is calculated based on the routing algorithms (in the case of traditional networks, Bellman-Ford, Dijnstra) that is prone to configuration by a network administrator via an user interface that permits vendor-specific commands and software via CLI. This is one of the main disadvantages of this technology as the device is completely subordinated to the vendor and the interoperability with other brands can be seen as a problem.

Nevertheless, regarding the right-side picture illustrated in fig. 2.2, the network devices become merely forwarding devices (mentioned earlier as part of the data plan) not taking any intelligence towards the incoming packets and subjected to the orders of the SDN controller (mentioned as data control plan) which carries a Network Operating System (NOS) that contacts core service. The communication between the controller and the network devices is interfaced employing Southbound API and between the controller and the network applications is via Northbound API [3].

Linking the definitions presented in the chapters above, SDN is launched based on the concept from software programming called abstraction to deliver such standardized outcomes-based on the fact that it is impossible and too complex to manage many low-level devices properly. Specifically, three kinds of abstractions are the fundamental basis of SDN [2]:

- **Forwarding**
  Forwarding provides all the information to the controller at the same time that hides low-level, device-related data.

- **Distribution**
  Distribution manages the status of the forwarding devices, collected by statistics at the same time that creates the topology of the network based on a global view.

- **Specification**
  Through virtualization and programs such as Network Hypervisors can translate commands into the behavior expected from the operators. It is the layer responsible for understanding the abstract view from the controller and mapping it into a more specific configuration.

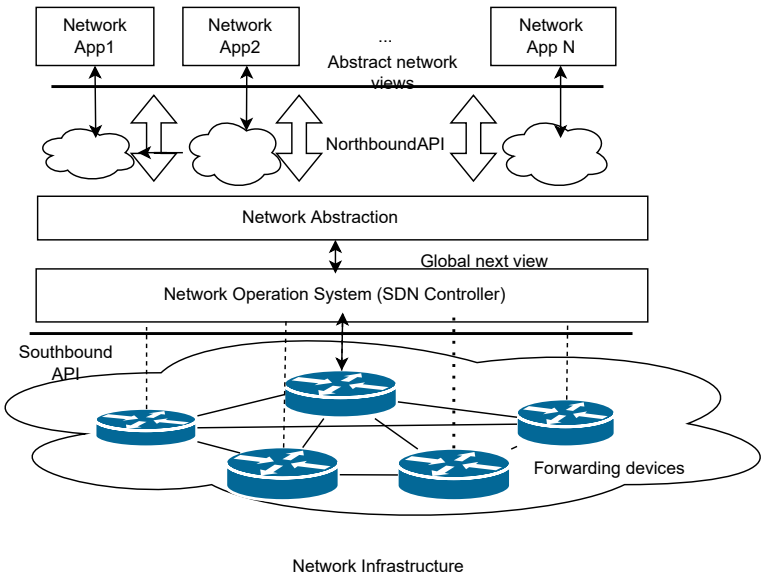


Figure 2.3: Overview of SDN abstractions.

Source: Stallings [5]

Consolidated these aspects, SDN architecture can be characterized by a balance of layers as per as the fig. 2.3. The concepts of control plane and data plane are similar to the previous definitions, with the introduction of the so-called Application Layer which is responsible for managing and configuring the network passing the input instructions given by the network administrator via Northbound API (normally REST API), the Southbound API (OpenFlow) and the Eastbound and Westbound API, in which exchange information between controllers. In some architectures can be placed Network Hypervisor or language-based virtualization.

Application plane

The Application plane is composed of network applications and services for control and management that work together with the control plane and the data plane. This layer is responsible for sending high-level commands from IT administrators into an instruction that will configure the devices in the data layer. It permits the IT staff to program the network to facilitate the deployment of services. The application programming can be implemented via API (Northbound Interface), script, and any format that allows the user to manage the network without knowing technical details. In other words, the application perceives an abstract view of the location of the network devices.

Given the nature of networks that could be indoor, outdoor, data center, enterprise and many others, appropriate applications must be developed to fulfill a business gap or to address an issue. There are several kinds of network applications and the most essential are grouped into five categories:

- Traffic engineering
- Mobility and wireless
- Measurement and monitoring
- Security
- Data centers

Examples of network applications could be Police Cop for traffic engineering.

One of the biggest issues related to Application Plane Security would be the vulnerability to malwares or any other malicious software. Furthermore, failures during the authentication and authorization can give access to illegitimate users the freedom to take advantage of the system and control the plane, as a consequence taking possession of the whole network.

Control plane

The Control Plane consists of a Network Operating System (NOS), known as the SDN controller, a software component that functions as the "brain" of the network. With the support of the Southbound APIs, it is possible to obtain an overview of the resources and the network devices without being limited to specific-commands devices. Furthermore, it is an important piece for the processing and then the creation of commands to configure the low-level device as per policy enforcement by the network administrator.

Overall, controllers provide an abstraction of the topology, and services as modules, and communicate with the business applications through the Northbound API. Examples of services are notification manager, a storage module, device manager, topology manager, and statistics manager. The basic flux of a new flow (Packet-IN) is developed in this way: the new packet is sent to the link discovery module to be analyzed, followed by the transmission of the link layer discovery protocol (LLDP) to the switches. Once finished with that part, the topology manager will get the acquired data in the previous process to calculate the network rote and save it into the database. In case of any changes in the network of topology such as link failures, after this process, the topology manager will recalculate the route and the database will be altered accordingly [7].

As the controllers manage the network devices, any security threat, malware, or issues during the carrying out activities can lead to disruption of the network in the case of the logically centralized the controller is critical as a single point of failure.

Depending on the topology, the network can be made of more than one controller. In a logically distributed manner, the controller is considered the central commander, while the other controllers are the subordinates grouped in a hierarchical cluster. Different from the centralized model, the distributed model offers resilience to failures. As an example, if one controller is down another one is in charge of its end nodes and manages them to avoid any congestion. Examples of centralized controllers are Floodlight, and OpenDayLight, while ONOS, Ryu, Disco, and HyperFlow are examples of distributed ones.

One of the biggest issues associated with this design is the fact that a potential attack on the main controller can compromise the whole network and a lack of synchronization of information via API (Westbound and Eastbound) could open the door for security breaches. Because there is a mismatch of updates between the controllers, the property of information consistency must be affected. As an example, if network status changes, the information is not propagated to all controllers and application firewalls may have information misled in their system [8].

Another important aspect to be discussed about the controllers is the adoption of network hypervisors that guarantee the specification abstraction. By definition, hypervisors are pieces of software that enables resource sharing of a machine such as RAM, CPU, and Network between multiple users at the same in a completely isolated way. The SDN Hypervisors are placed between the application and

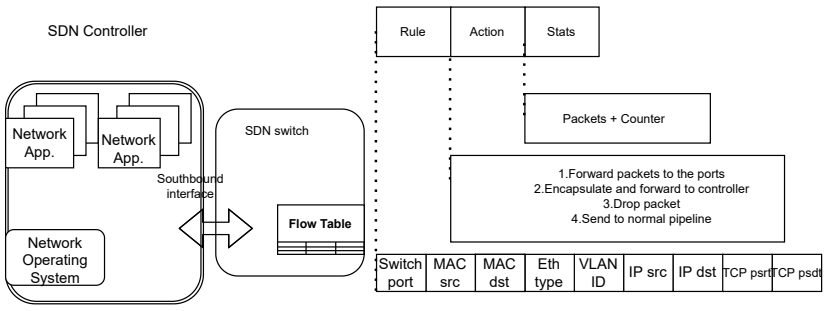


Figure 2.4: Flow table scheme.

Source: Stallings [5]

the control layers, providing isolation between users and isolation of the data. Indeed, they are responsible for transforming the network global view of the network topology into an abstract view. In other words, instead of modeling the network topology as a gigantic graph view, like edge and nodes, the network topology is seen as a "big switch" where the Hypervisor would be represented as the central node and the edges would be ports of that switch[3].

Data plane

The Data Plane layer consists of the hardware components such as routers and switches that transfer the packets in an SDN network to their destination, working as subordinates to the controller which sends forwarding rules to update, modify or delete network routes according to the demand. This layer is divided into as well as exchange traffic information. The data plane is divided into: control flow and data flow.

The control flow is positioned between the OpenFlow switch and the controller and is responsible for carrying out the actions defined by the controller, exchanging statistics of the connection and other signaling messages via the Southbound interface. It is protected by SSL/TLS authentication protocol and it is essential for the communication of the compatibility of the switches and the controller with the protocol OpenFlow to guarantee the interoperability of the equipment involved in the data transmission.

The data flow is the part where the switch receives the packets and stores them into the input buffer. There waits in queue to be processed until being matched with a rule and then applied to the correct action according to the flow table. After that, the packet is discharged at the output buffer to be sent to its final destination. In summary, it is a part that only deals with data packages.

The OpenFlow-enabled switches are made of a sequence of flow tables that comprises a matching rule, an action to be performed on the packets, counters to produce the statistics of the matched packets to the controller, and the hard timeout and idle timeout. Respectively the maximum duration of a rule storage in the flow table regardless of inactivity of the traffic and the maximum duration of a flow rule in the flow table without activity. Some of the basic features are exemplified in fig. 2.4.

OpenFlow switches identify traffic according to the notion called "flows" which is a sequence of packets transmitted over the network that share some header fields, for example, packets that have the same IP destination, IP source, transport protocol, MAC origin, and destination. When a packet arrives at the entry port of a switch, it will be compared with the rules that belong to a flow table and in

case of a match, proper action will be endorsed. Actions are activities to be performed on the packets that could be to drop the packet, forward it to a specific port, or modify the packet headers according to the instructions from the controller, such as encapsulating and forwarding to the controller and sending to the normal processing pipeline.

Essentially, the packet arrival is handled as the following sequence:in the case of a packet that matches two different flows, the output is the flow rules with the highest priority according to the policies given by the administrator. After the match, counters related to that flow will be updated such as the number of flows and numbers of bytes. Finally, the action will be applied accordingly such as flooding the ports or dropping the packet (if the instruction to drop the packet is given by the controller).

When a flow is not identified in the forwarding table, it is considered a table miss, and three kinds of outcomes occur: drop, send to the controller and send to another flow table in the same switch. In the case of sending to the controller, the packet is encapsulated and forwarded in a form of Packet-IN messages. These messages are a request from the switches to calculate the best routes using the controller topology discovery module. Once calculated the routes, a message is transmitted to the switch to create a new flow rule that will populate the flow table called FLOW-MOD. As a consequence, the next arriving flow will be processed faster without passing by the controller.

To perform all this processing, OpenFlow-enabled switches are equipped with memory fast enough to store and manipulate the flow rules. However, this memory presents drawbacks such as high financial and energy consumption and capacity limitations. The limitation results in vulnerabilities that lead to Denial of service attacks such as saturation attacks and , exhaustion which will be mentioned in the next chapters.

Northbound interface

Northbound interface is an API located between the application plane and the control plane, allowing administrators to control and manage the network hiding more complex details. The standardization of this API is still in process so it is still difficult to achieve interoperability between the controller and the application. Normally the API used is REST. This API is responsible for translating application requests into low-level instructions to the network devices and visualizing in a "user-friendly" way the statistics provided by the network devices and processed by the controller.

Southbound interface

Southbound Interface is an API located between the controller and the infrastructure layer. It transmits the instruction to the network devices such as blocking a specific flow, changing routing path, flux control, etc. These instructions aim to program the flow table of a switch as was previously explained in the data plane section.

Therefore, the API transmits the information from the devices to the controller in the following cases: occurrence of modification events of a door or link, the transmission of statistical data (providing detailed information to the network administrators), and packet transmission to the controller in an event of table-miss or simply when action registered in the flow table request to send the information to the controller.

In "Software-Defined Networking: A Comprehensive Survey", it is mentioned other APIs such as

OVSDB, ForCES, and OpFlex apart from OpenFlow. OVSDB provides extra functionalities that can be complementary to OpenFlow [2].

2.2 SDN Application services

SDN facilitates the implementation of mechanisms such as tunneling, routing, and load balancing regardless of the environmental conditions such as home, enterprises, indoor or outdoors. For instance, performance analysis of a topology or architecture can require knowledge of a vast number of protocols and hardware, each one associated with a specific vendor could be a costly endeavor in terms of time and effort. Not mentioning the issues related to inconsistencies of data originated from diverse factors. Considering this scenario, SDN applications can manipulate the network dynamically which guarantees flexibility and fewer issues with the interoperability of devices. D. Kreuz et al.[2] grouped the SDN network applications according to the following categories :

Traffic Engineering Traffic Engineering aims to optimize network performance by modeling, measuring, classifying, and controlling traffic. To provide efficient operations, while optimizing resources and performance. D. Kreuz et al.[2] cited some applications solutions available in the market such as load balancing, rules placement, techniques for flow management, flow tolerance, topology update, traffic characterization, and traffic optimization.

Mobility and Wireless Wireless and Mobility networks introduce a new set of features to be addressed by the control plane. However, SDN design facilitates the deployment of these features in the form of SDN Applications. Some of the existing application solutions consist of spectrum management, resource block allocation, interference management, handover mechanisms, control and transmission, power management, load balancing provision etc. Indeed, some applications facilitate the deployment of heterogeneous technologies, interoperability between networks, and access control[2].

Measurements and Monitoring D. Kreuz et al.divided Measurement and Monitoring into two categories : applications that add new functions to network services and applications that improve the metrics of a SDN network. The first can be exemplified by the implementation of measurement solutions to a SDN-based broadband home connection. The second option is related to techniques to mitigate overloading the controlling during the gathering of statistical information from the data plane. These techniques include deterministic and stochastic sampling techniques, traffic matrix estimation,two-stage Bloom filters, fine-grained monitoring of wildcard rules etc [2].

Security D. Kreuz et al.[2] categorized SDN application security into applications aimed to protect SDN from its vulnerabilities and applications aimed to reinforce SDN network security as a whole. Most of the options utilize SDN as a mechanism to mitigate or detect security issues in network systems such as policy enforcement, DDoS attacks detection and mitigation, random host mutation, traffic anomaly detection, etc.

Data Center Networking SDN stands as a technology to solve the challenges related to efficient management of data centers. The benefits can expand to live network migration, failure avoid-

ance, troubleshooting, optimization, provisioning of middleboxes-as-a-service, Qos support, real-time detection of anomalies, etc [2].

2.3 Security advantages and disadvantages of SDN

In terms of security, Software Defined Networks present common security challenges in comparison to the traditional networks. However, its unique architecture comes with specific challenges and presents benefits and drawbacks with respect to traditional networking architectures.

Advantages Some of the advantages include the efficient monitoring of abnormal traffic and rapid response to threats.SDN controllers can monitor the whole network at the same time and visualize and monitor anomalies on run-time. Furthermore, during an attack event or detection of a traffic abnormality, IT staff have a possibility of mitigating the impact immediately [9] [2].

Disadvantages The disadvantages include the centralization of decisions at the controllers that places the device as a vulnerable piece, the utilization of API, and the introduction of new attacking points. SDN is characterized by the logically centralized controller that accumulates functions such as network visualization, control of the network devices, and hosts the application modules such as routing, firewall, or load balancing. In case of a single failure or attack, the impact affects the whole network [9].

The fact that SDN is based on API that facilitates the discovery of failures and vulnerabilities to be exploited by attackers. Furthermore, it facilitates the deployment of malicious codes. Lastly, SDN introduces more attack points concerning traditional networks such as: OpenFlow switch, the channel between switch and controller (Southbound API), controller, channel between the controller, and the application plane(Northbound API), and the applications [9] [2].

2.4 Related Works

The related work section is organized according to table 2.1 it is divided into the category the Nature of the survey: General, Attack specific, or Plane specific. General is the label chosen for the survey that covered all the layers of an SDN architecture. Attack specific simply refers to the surveys dedicated to an attack. By logic, plane specific is dedicated to just one attack.

Table 2.1 Table presents some remarks such as the article [13] and [7] do not present a solution to a specific attack, for that reason was labeled as "Not specific". The number of attacks referenced were extensive that could not be placed in this Table.

Kreutz et al. [2] highlighted vulnerabilities exclusive to SDN architecture such as exploitation of the controller and the deployment of malicious applications at the Security session. Followed by citation of security issues in Open-Flow networks and using the STRIDE model (Spoofing, Tampering, Repudiation, Information disclosure, DDoS, and elevation of privileges). The article discussed the origin of these vulnerabilities and provided recommendations to mitigate these threats. Among the presented solutions are accessed control, flow aggregation, attack detection, event filtering, firewall and IPS, forensics support, etc.

Article	Nature of the survey	Description of the survey	Security issues or attacks
[2]	General	Comprehensive survey of SDN architecture covering a range of topics not directly correlated such as scalability, hardware, etc.	STRIDE model
[10]	Attack Specific	Quantitative review of existing DoSand DDoS solutions	DoS and DDoS
[11]	Attack Specific	Survey of topology related attacks in SDN	Topology attacks
[12]	General	Survey of SDN security solutions related to the three planes: Data Plane, Control Plane and Application Plane	AAA, DoS, malicious flow rules, flooding attacks, controller hijacking, MiTM and TLS authentication
[13]	Plane specific	Survey of control plane issues focused specifically on the control architecture, performance, scalability, placement, interface and security	Not specific
[14]	Plane specific	Survey of application plane and Northbound interface security issues and solutions	AAA, illegal function calling, trust authentication, malicious flow rule injection, DDoS.
[7]	General	Security survey of SDN in different perspectives including the categorization of attacks and solutions divided in three plane	Control plane:Host Location Hijacking, link fabrication, port amnesia, traffic sniffing. *
[15]	General	Survey of recent trends and new challenges in SDN	Not specific
[16]	General	Survey of security issues and attacks in SDN	Unauthorized Access, Data Leakage, Data Modification.
[17]	General	Survey of security issues and attacks in SDN separated by three planes: Data Plane, Control Plane and Application Plane	AAA, DDoS
[18]	General	Survey of security issues and attacks in SDN separated by three planes: Data Plane,Control Plane and Application Plane	Malicious app, AAA, flow conflict, DoS, side channels, hijacked

Table 2.1: Categorization of existing surveys according to nature and security issues or attacks.

Aladaileh et al. [10] surveyed DDoS and DoS detection mechanisms available in the literature. Based on extensive research, the author organized the studies into a qualitative comparison of the articles investigated according to the following detection techniques: entropy-based, low rate traffic, time-based, intrusion detection, and machine learning. Furthermore, the authors organized the detection mechanism in the following criteria: techniques and features, threshold, and the location of the SDN environment deployment. The organization of DDoS detection mechanisms into detection techniques was not a novelty considering the number of articles dedicated to these solutions and it was used as a base for the DDoS solution section.

Khan et al. [11] surveyed the topology discovery process and potential threats to organizing a taxonomy of topology attacks. The authors organized attacks into host-based or switch-based attacks, controllers vulnerabilities, solutions, and other threats that were considered consequences of topology attacks. Regarding the control plane, the authors mentioned the lack of authentication of the Packet-In messages in Host-Tracking systems and to obtain the origin of LLDP packet during the Link Discovery Procedure.

Ahmad et al. [12], the authors discussed the problem of application authentication and authorization of network resources. Hence, it highlighted the issue of extrapolating the link capacity of a controller during a Packet-in event and the number of switches that can handle common harm. Lastly, managing distributed controllers and facing possible DoS and DDoS were cited among other threats.

Previous studies were launched regarding security at control planes centralized in some aspects such as Architecture, performance, scalability, placement, and interface with little priority on security. Xie et al. [13] addressed the data integrity, confidentiality,and authentication among controllers and the development of Ethane to authenticate new end nodes added to the network. Furthermore, it was mentioned the efforts to avoid rules conflicts in flow tables and the development of a security application framework called FROSCO to guarantee controller security. There was a small piece dedicated to the security topic since the article was not emphasizing this topic.

Rauf et al. [14] perceived that most of the security surveys involving SDN were focused on control and data planes, realizing that few efforts have been made toward Northbound Interface vulnerabilities. Given the fact that any misuse or malicious application in this layer can directly impact the functionality of the control plane, such as illegal access and exposure to malware are one of the threats. Due to this dependency between the application layer and controller, the Northbound threats and vulnerabilities were added to this work.

Rahouti et al. [7] analyzed security attacks by layers and their implications. In the case of the control plane, it was mentioned host location hijacking, link fabrication, and man-in-the-middle to name a few. In later sections, the authors surveyed the SDN security frameworks to solve gaps and point out the challenges associated with their implementations. Among these solutions, Floodligh-SE and DDoS threat detection using Self-Organizing Maps (SOM) was already extensively mentioned in other surveys.

Ochoa-Aday et al. [15] cited the weakness of controllers such as Beacon, Floodlight, Maestro, OpenDayLight and Pox are prone to DoS, repudiation, spoofing, etc. The author cited as potential solutions controller protection by the utilization of middleboxes between the control and data plane, controller extensions, and trust authentication methods.

Scott-Hayward et al. [16]'s article was organized into main security issues such as Unauthorized Ac-

cess, Data Leakage, Data Modification, Malicious compromised applications, Denial of service, configuration issues, and system-level security. Based on this categorization, the work was developed by mapping the existing surveys and the corresponding impacted layer or interface. The later chapters were a discussion of improvements and future directions.

Another type of survey of categorization of security issues was done by Jiménez et al. [17] diving into the security issues by blocks: Northbound interface and application plane, East-Westbound interface along with control plane, and Southbound interface with the data plane. For each block was surveyed the security issues, solutions, and classification according to the STRIDE model.

Liu et al. [18] presented another alternative categorization survey by separating the five layers and mapping the security issues. Northbound and Southbound sections were briefly cited and not a specific issue was emphasized in these sections.

All the review articles surveyed presented their perspective of security attacks according to the scope. Specifically, the security articles are constrained to cover a layered attack or the most frequent attacks and issues. None of the chosen articles introduced novel attacks out of the traditional ones. Considering the gap of surveys dedicated exclusively to the SDN controller, this Thesis was created to cover fulfill this void and provide a new perspective on SDN control plane security.

2.5 CHAPTER 2 CONCLUSIONS

This chapter introduced the basic concepts of the SDN architecture along with the advantages and disadvantages regarding this technology. In the next chapter, it will be introduced the attacks and security issues relevant and their definition and concepts that are of extreme importance for this Thesis.

3

Security at SDN control plane : State-of-the-art and Challenges

This chapter introduces the attacks and security issues relevant to this Thesis categorized according on their impact to the SDN control plane. This chapter is the backbone of this work to comprehend the next sessions.

3.1 SDN security challenges

Before digging into the core of this work, it is important to define correctly the concepts that will be frequently used in this survey.

According to NIST, vulnerabilities are defined as "Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source " [19]. Clearly, system weak points opened to threats. Examples of vulnerabilities are non-updated software and a login page that permits access to unauthorized users.

A threat is defined by NIST as "Any circumstance or event with the potential to adversely impact organizational operations (including mission, functions, image, or reputation), organizational assets, or individuals through an information system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service. Also, the potential for a threat source to successfully exploit a particular information system vulnerability"[20]. In other words, it is the condition to suffer a potential violation of the system. Threats could be accidental or intentional such as a buggy software or a MitM attack, respectively.

On the other hand, attacks are defined as " Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself " [21] by NIST . In summary, it is the successful execution of an intentional threat exploring a system vulnerability.

SDN is prone to threat vectors (the way the attack is carried out) and vulnerabilities, some of which are not exclusive to this technology and are still present in legacy networks. Indeed, SDN opens a path to new weak points due to its new architecture and the introduction of a new player, the Openflow

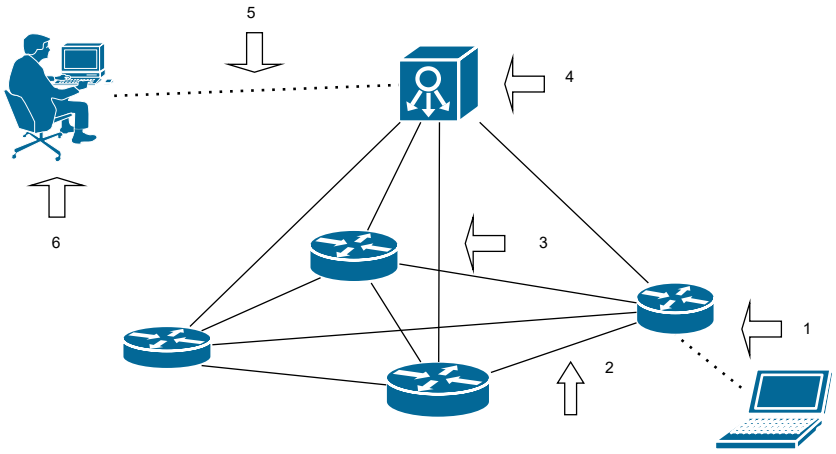


Figure 3.1: Attack position in SDN

Source: Z. Shu [9]

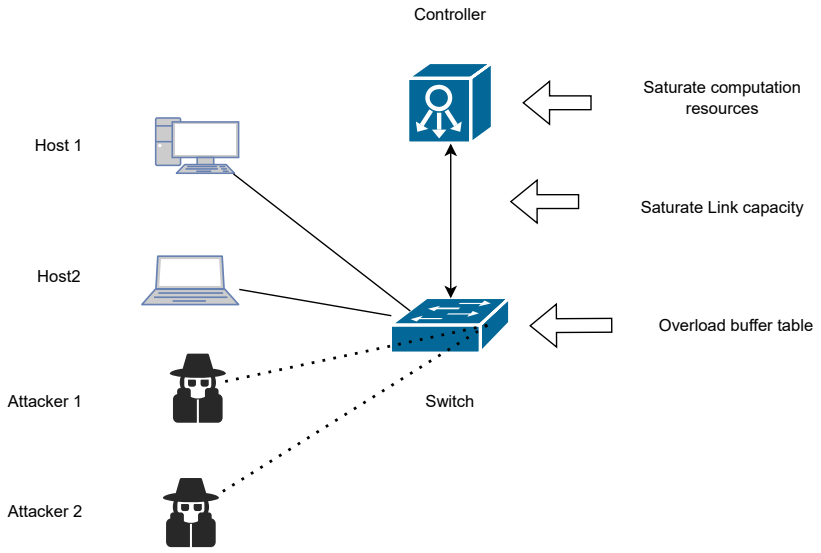


Figure 3.2: Generic attack scenario

Source: J. Surya Maddu [23]

protocol, responsible for the exchange of data between the controller and the data layer [2].

Some of the points to pose an attack are highlighted in fig. 3.1 is represented by the numbers and arrows. The first arrow illustrates attacks targeting switches such as the Flow Table Entry attack [22] along with other types of saturation attacks. The second arrow shows a candidate point to the link between switches that are not encrypted [9]. The third arrow points to the channel between the control and the switches, which is vulnerable to eavesdropping and MitM [9]. The fourth arrow points to the controller, vulnerable to attacks such as DoS and Topology attacks. The fifth arrow targets the link between the controller and the application plane, this channel is exposed to the use of Malicious and flow injection. The sixth arrow points to the working station contaminated by viruses, without proper authentication and access control can affect the controller [2].

DoS attack can take down the whole network such as any exploitation of the controller functionalities, its resources such as memory, CPU, and the deployment of malicious applications. Indeed, the novelty of the softwarization of networks led to the introduction of exclusive threats to applications

such as lack of proper authentication, wrong authorization or access, or intrusion of malware. Any vulnerability that affects the controller, as a consequence affects the whole network system due to its design dependency.

3.2 Classification of control plane attacks and threats

This section aims to facilitate the comprehension of attacks and the classification as direct and indirect attacks. The simple definition impacts the controller, whereas indirect attacks impact other components, but the outcomes affect the controller in the end.

To visualize this idea of the direct and indirect impact, fig. 3.2 shows an example of DDoS attack targeting different points of an SDN architecture. Modeling the attack at the respective points as "cause" and "consequence " can help to analyze the proposal of this selected categorization. It can be inferred the attack point leads to a "consequence" of attacking the control plane. Analyzing the figure from the bottom to top, an attacker targets to overload the buffer memory of a switch with spoofed IPs as a "cause ".The " consequences " would be the transmission of Packet-in messages to overload the controller bandwidth capacity and saturate the resources. It is noticeable that not attacking directly the SDN, using an intermediate, is not sufficient to confirm that the controller is not under attack.

For this reason, the next subchapters will introduce the definition of each security issue and attack according to the categorization as directly or indirectly targeting the control plane.

The table fig. 3.3 and the mindmap shows the division and the groups and subgroups of security issues and attacks according to the categorization path chosen in this thesis.

3.2.1 Direct Attack

- **Spoofing**
  Spoofing is the act of misleading or falsifying information to impersonate a user and obtain privileges or to be the opportunity to launch a more dangerous attack such as Man-in-the Middle, Ddos. There are several types of spoofing such as: email spoofing , ID spoofing , IP spoofing, ARP spoofing, and DNS spoofing.

  E-mail is a recurrent threat in which the user receives an email from a sender that pretends to be part of an institution or trusted origin. This email seems typically legitimate and if it is mistaken by the user, it provides unauthorized access to his/her data. The most popular subgroup of these attacks is the ARP, DNS, and IP spoofing which are also legacy attacks from traditional networks. IP spoofing is an attack related to the capacity of the attacker of forging the origin IP field of a datagram to mislead users to trust in the authenticity of the source.

  ARP Attacks: ARP attacks are legacy from traditional networks and can be transformed into an open door for man-in-the-middle or DoS/DoS attacks. Protocol ARP is used for MAC address discovery when a host wants to send a packet to a destination of an unknown MAC address. Firstly, the sender consults its ARP cache table, in case the MAC address is not in this table, the host sends a broadcast packet called ARP Request. If the destination is inside the LAN, the destination responds with an ARP Reply and its information is saved into the ARP table of the initial host. In case the destination is not on the same LAN, it is obtained the MAC address of the default interface that will route the packet [64].

Impact controller	Category	Security issue and attacks	Description of the Attack	Articles
Direct	Spoofing	ARP Spoofing	Falsifies an information to deceive the victim to be legitimate	[8, 24, 25]
	DDoS	Application Layer	Overload a system or server to turn it unavailable to the legitimate user.	[26, 27, 28, 29, 30]
		Flooding		
		Amplification		
	Northbound interface	Access control	Specify permissions to access system resources	[31, 32, 33, 34]
		Authentication and Authorization	Process of confirming the validity of credentials	[35, 36, 37, 38, 39]
		Malicious application threats	Usage of infected software to damage other systems	[40, 41, 42]
	Topology attacks	Host Location Hijacking	Host Tracking system poisoning	[43, 44, 45, 46, 47]
		Link Fabrication	LLDP packets poisoning	
	Others	Fishing	A social engineering attack to steal personal data	[48, 49]
		Personal Hijacking	A binding attack intend to steal's the victim IP	
Indirect	Fingerprinting		Gathering information to profile the user	[48, 49]
	Southbound interface	Fishing	A social engineering attack to steal personal data	[50, 51]
		Authentication	Process of confirming the validity of credential in TLS	[52, 53]
		Man-in-the middle (MitM)	Interception of a communication	[54, 55]
		Data plane saturation	Flow table overloading	[56, 57, 58]
	Reconnaissance and port scanning	Reconnaissance	Process of gathering victim's information to launch an attacks	[59, 60, 61]
		Port scanning	Message transmission to detect door vulnerabilities	
	Side channel	Timing probings	Collection of intrinsic information to launch an attack	[62, 63, 14]

Table 3.1: Control Plane Attacks

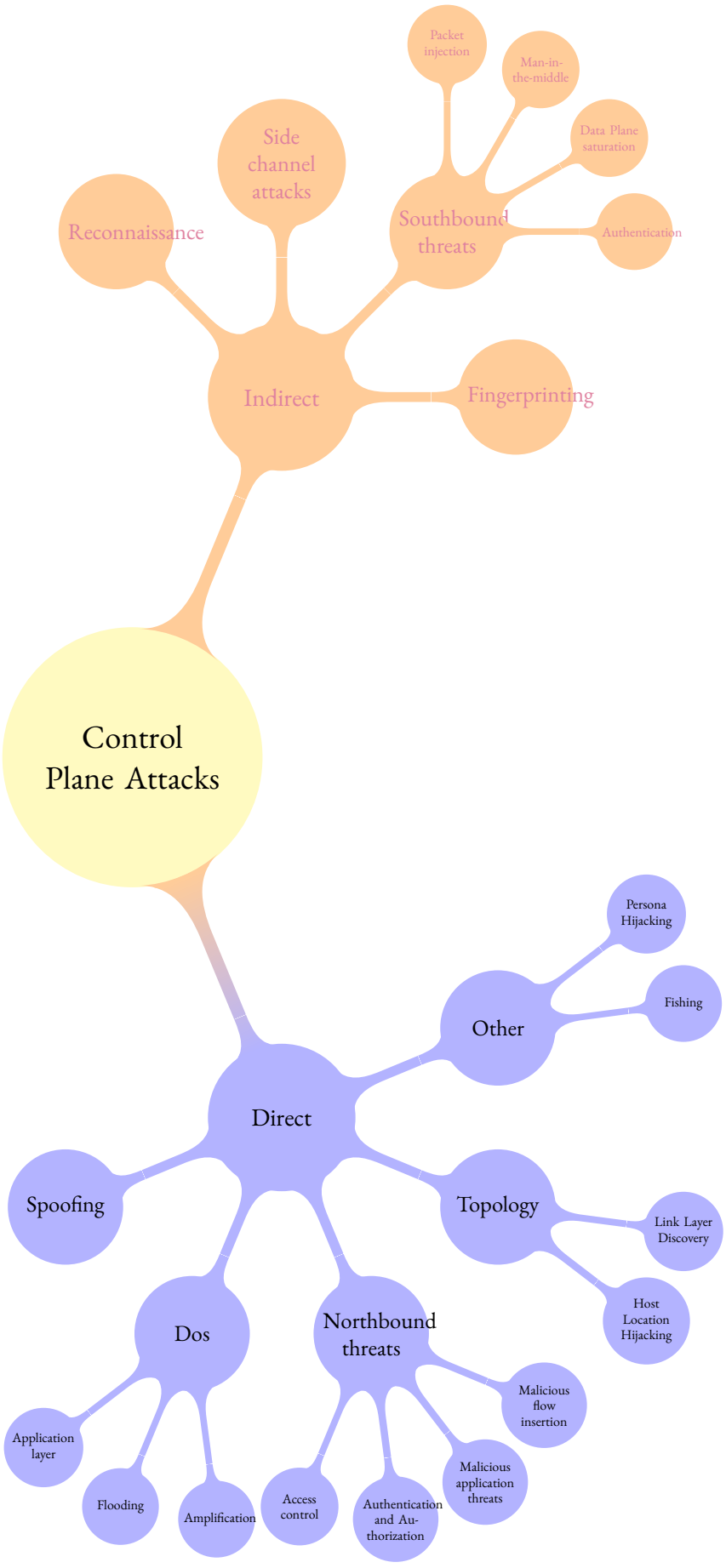


Figure 3.3: Control Plane Attacks

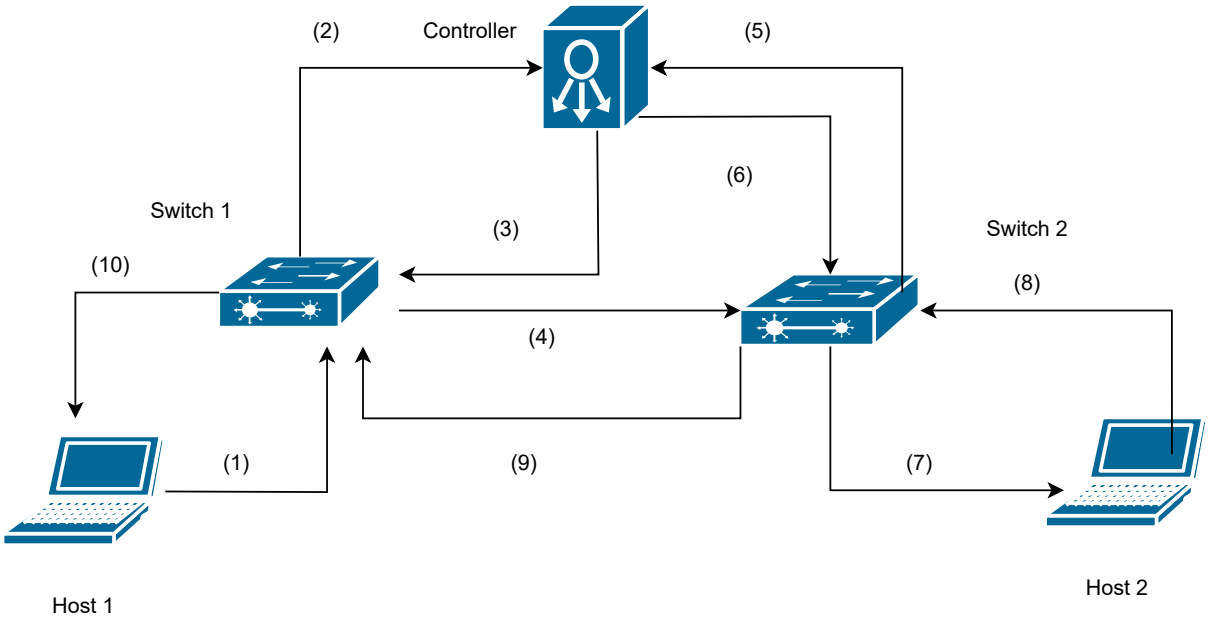


Figure 3.4: ARP scheme

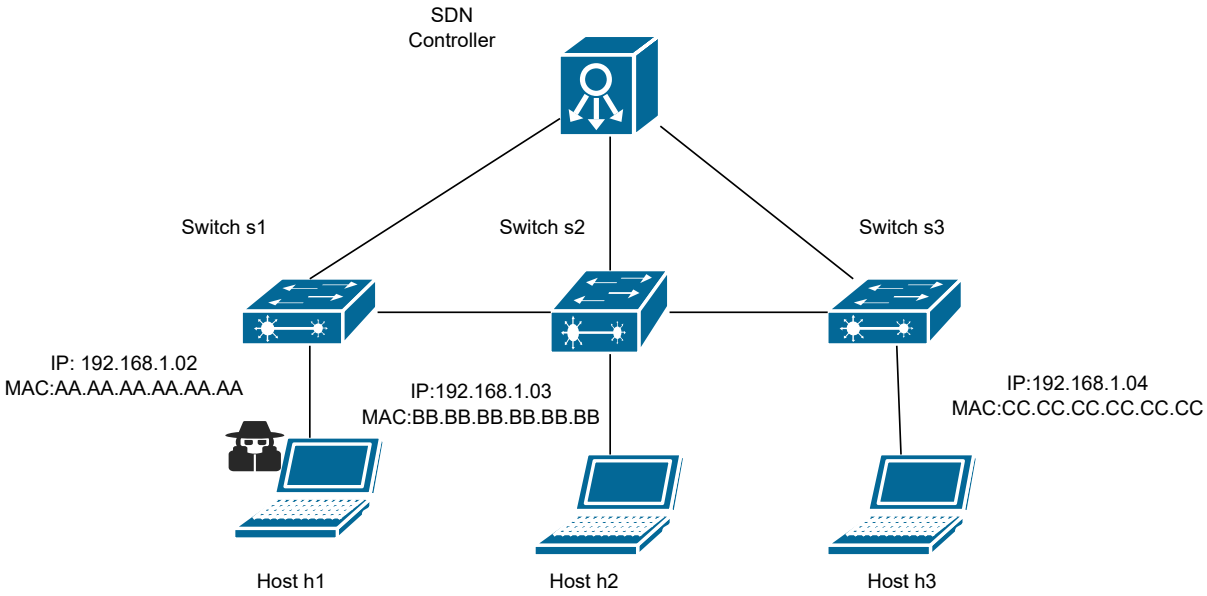
Source: Zhang et al. [66]

There other types of ARP packets such as ARP probes that are sent in broadcast to verify if an IP address is available to use in the network. In case, the IP is available an ARP Reply is sent back to the original sender in unicast. If there is no response back, an ARP announcement is sent in broadcast mode from the sender to claim its IP address [65].

Considering the implementation in an SDN network, the basic steps of an ARP protocol are illustrated in fig. 3.4. Firstly, host 1 sends an ARP Request to switch s1 to obtain the MAC address of host 2 (1). In the case there is no match in the flow table, the packet is encapsulated and sent to the controller as a Packet-In message (2). Controller retrieves information such as IP address, MAC address, and location of host 1 and sends back a Packet-out message to switch 2 to transmit ARP Request (3). In the next step, an ARP request is sent from switch 1 to switch (4). In the case there is no matching rule, the switch sends a Packet-In to the controller (5). The controller sends a Packet-Out to switch 2 exactly as step 3 (6). Finally, ARP Request is sent from switch 2 to the destination host 2 (7). Host 2 sends back an ARP response to switch (8). In the case switch 2 has a matching rule, the packet is sent directly to switch 1. Otherwise, the process of forwarding Packet-In to the controller is repeated to obtain the information of the distance location of switch 1 along with the distance location of switch 2. Then the controller sends the flow rule to switch 1 and switch 2. In the next step, switch 2 transmits the ARP Response to switch 1 once the matching rules are updated. In the end, switch 1 in possession with the location of host 1 transmits the packet (10) [66].

ARP Spoofing During the ARP spoofing, promiscuous packets are generated so the attacker impersonates another host to poison the victim's ARP table and obtain direct exchange communication. The idea of ARP spoofing in traditional networks is very similar to ARP spoofing in SDN and it will be explained in the scheme [65].

A host responds to a request impersonating the IP address of another machine, once it is not verified the authenticity of the packet in the datagram, this host poisons the ARP table of the requesting host. Finished this process,this host can exchange packets with the requesting host



Source: Alharbi et al. [24]

Figure 3.5: ARP spoofing scheme

in the name of the victim's host. For example in fig. 3.5, host 3 manages to communicate with host 2 and sends an ARP Request in broadcast all the interfaces (the example is not considering the interface between controllers and switches) to discover host 2's MAC address which is absent in its ARP cache table. Host 1 forges the origin IP field as `192.168.1.03` and origin MAC address field with `CC:CC:CC:CC:CC:CC`. Without proper authentication, host 3 understands the packet as legitimate and the IP-MAC association as valid. As a consequence, the next attempt from host 3 to host 2 will be redirected automatically to host 1 [24].

- **Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks**
  Denial of Service (DoS) and Distributed Denial of Service (DDoS) are attacks that target to render a service or network and computation resources such as bandwidth, CPU, and memory unavailable to the legitimate user [67]. It is considered one of the fearest attacks by companies and organizations. In February 2018, the platform for software development GitHub suffered an amplification attack memcached server with UDP using port 11211 [68]. The amplification attack called "mencrashed" misleads memcached servers spread around the globe, using basic techniques such as IP spoofing to obtain bigger responses than the capacity of the victim, in this case, GitHub.

  DoS and Distributed Denial of Service (DDoS) are different concerning the number of hosts involved during the attack. While a simple DoS attack is performed by a single machine to flood the victim, DDoS is an extended version of DoS using a large number of hosts as it is exemplified in fig. 3.6. Firstly an attacker penetrates a host and installs malware to obtain total control of the machine that becomes its slave. Along with other vulnerable victims, it is formed the army of bots that only replicates requests and orders from its bot master. The army is called botnet [69].

  Even though it is considered a legacy attack, an extensive literature has been released regarding DoS and Ddos attacks in SDN. These attacks affect the controller, switches, and interfaces. However, it is the control plane the most critical point once it concentrates all the routing decisions. In SDN, spoofed packets are sent in large numbers to the switches that after consulting the flow table can't find a flow match. As a consequence switch send a Packet-In message to the

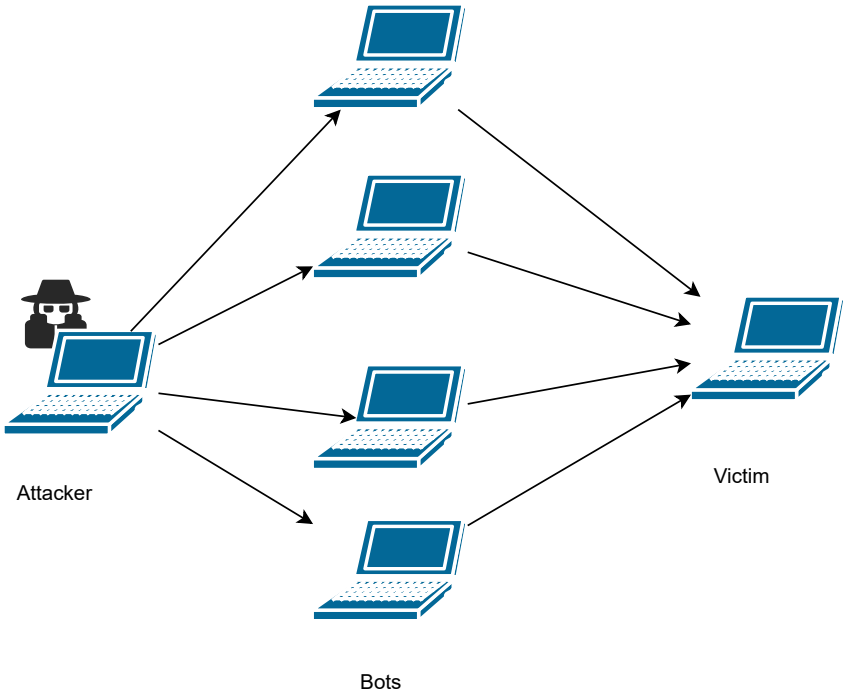


Figure 3.6: DDoS scheme

Source: Aljuhani [69]

controller to create a flow rule. As these operations are carried out by a botnet, the capacity of the controller is extrapolated.

This work divides the DDoS attack into the following categories: application-layer attacks and volumetric attacks, subdivided into flooding and amplification attacks [70]. Volumetric attacks are the most common and exploit vulnerabilities in the third and fourth layer, flooding the network with a high volume of requests and high throughput.DNS Amplification, Amplification, Flood, and SYN Flood are some examples. In other articles, it can be found other subdivision such as volumetric and protocol attacks, etc and attacks that could belong to more than one category at the same time such as HTTP Flood could be also grouped as a volumetric attack. For didactic purposes, this categorization was chosen to highlight the layer stack and intrinsic characteristics of the protocols.

Application layer attacks These attacks overwhelm the resources of a web server with HTTP requests mimicking the behavior of legitimate users. As the name suggests, they are launched in the application layer stack. The most common examples include Slowloris, HTTP-GET Flood, HTTP-POST Flood, SMTP, and others. Bots are ordered to establish a HTTP Request connection with the web server until reaching its maximum capacity. As a result, the service is unavailable to handle legitimate requests.

At certain points, HTTP flooding attacks diverge from other flooding attacks in a more sophisticated knowledge by the perpetrator such as establishing a TCP session before the actual attack, use of legitimate IP address, isolation of the application layer stack so anomalies can't be detected by the lower layers and imitation of user's behavior by the bot machines [26]. Tools like HTTP Unbearable Load King (HULK) and High Orbit Ion Cannon (HOIC) are options of HTTP flood attack tools available.

A basic HTTP flood is a seventh-layer attack and it is common usage is to load webpages or transmission of contents to forms.Normally, HTTP Flood attacks are classified into: HTTP

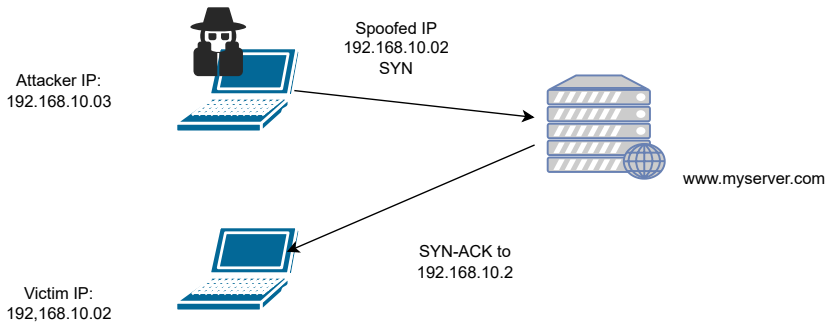


Figure 3.7: SYN Flood scheme

Source: Aljuhani [69]

Get an HTTP Post. In the first attack, webserves receive a huge volume of requests to send images, files until the target service is denied to the legitimate users. The second type, the POST method is used to register users and fulfill HTML forms related to complex server processing such as database access. This overprocessing is the key to consuming resources until the disruption of the service.

Another type of application layer attack is the Slow HTTP which includes Slowloris and Slow HTTP Post flooding attack. The idea is the same as HTTP Flood. The attack is to use an HTTP Request with an incomplete HTTP header. Upon receiving the uncompleted packer, the webserver does not refuse the connection, counting the fact the HTTP packet will be sooner completed. The accumulation of incomplete HTTP requests turns the connection to the web server unavailable. This attack is especially hard to be detected since that it is similar to a user's behavior [27].

Flooding Attacks Similar to the explanation of HTTP Flooding attacks, generic flood attacks are released using an army of bots to exhaust the resources of the target. Examples of flooding attacks are Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) and also Hypertext Transfer Protocol (HTTP) were mentioned in the previous section but it was classified as an application attack due to their complexity to be launched. The low-layer flood attacks can be simply launched by user-friendly tools such as hping3 without any advanced security knowledge.

The process of TCP SYN attacks is explained as follows according to fig. 3.7. An attacker abuses the three handshaking processes to obtain advantages. In a three-way handshaking processes a host expresses the desire of communication with a server in a given port, for a web server is port 80 with SYN flag active. The server responds with an SYN plus an ACK to acknowledge the previous packet and waits for the completion of the connection if it is not timeout. In other words, the server waits for an ACK from the sender. During the attack, once it is discovered that the server's port is open the attacker will try to establish a connection using the SYN flag with a spoofed IP. As a consequence, the server's response containing the SYN-ACK will be redirected to the client whose IP is the same as the spoofed one. In the meantime, the server is in half-opened state waiting for the acknowledgment, the memory research is not released until receiving the acknowledgment back. If this attack is carried out by a botnet the server is pushed to exhaustion and starts dropping legitimate connections [71].

UDP Flood diverges from the TCP flooding in the absence of the three-way handshaking connection.The idea is not to let the server be in the half-connection state waiting for the acknowledge. Instead, the objective of this attack is to occupy the link capacity of the server. This attack is initiated using a great amount of UDP packets to random ports on a server that is not obtain-

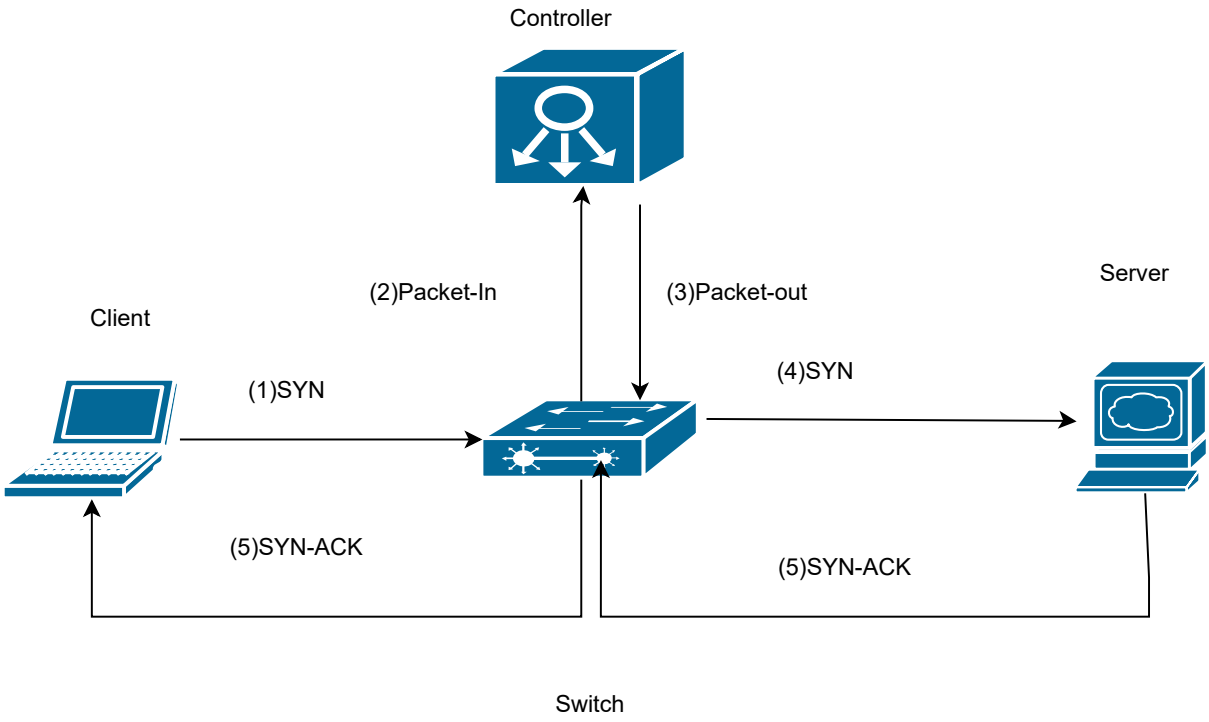


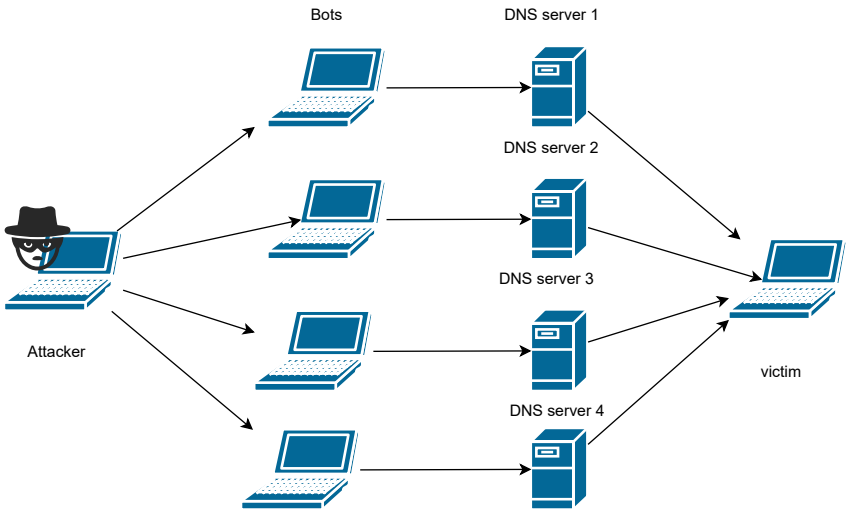
Figure 3.8: SYN Flood scheme in SDN

Source: Aljuhani [69]

ing the application that is listening to this port, and sends back an ICMP message of destination unreachable. Transmitting large amounts of UDP packets with spoofed IP at high rates, forces the response of the server of ICMP packets while consuming its resources. Again turning the server inaccessible to legitimate users.

In SDN networks, SYN attacks evolve in a slightly different way. fig. 3.8 illustrates this process given the following topology. An SYN packet is not matched with any of the rules in a switch flow table (1). As a standard procedure, a Packet-in message is redirected to the controller (2), until after the processing a Packet-out is sent back to release the switch buffer space (3). In the next step, the packet is finally sent to the destination server (4). The Server responds with an SYN-ACK and is currently in a half-opened connection state, waiting for the sender's acknowledgment(5). After being sent back to the switch, again it is verified if there is an existing rule sending the packet backward direction from the switch to the client. In case there's no existing flow rule, again it is repeated steps (2) and (3). Upon, the Packet-out is received or the matching rule is found, the switch transmits to the client the SYN-ACK packet (6). Considering that this is a long process, the server's memory is not released while waiting for the sender packet with an ACK flag. In case an attacker sends a great number of SYN packets with spoofed IP, it can either exhaust the switch's input buffer or occupy the bandwidth between the controller and switch [72].

Ping or ICMP flood attack is characterized by sending a great number of echo requests with spoofed IP to a target server. Once the echo request is received at the server, it is expected to respond to each request with an echo reply which demands processing from the server. When an attack is orchestrated with a huge number of hosts using echo requests that are beyond the server's capacity, the victim turns unresponsive or unavailable.



Source: Aljuhani [69]

Figure 3.9: DNS Amplification attacks

Amplification attacks As the name suggests the idea behind this kind of attack is a small or even harmless small packet size is sent but the response towards the victim is bigger than the original content of the attacker. The content is magnified (amplified) until taking down or degrading the victim's resources. In simple words, it is like giving a long response to a simple straightforward question.

An attacker orchestrates bots with a source spoofed IP into a messaging broadcast IP address in the subnetwork. As a reaction, the broadcast IP will reply to the victim. Examples of this kind of attack are Smurf, NTP Amplification, Fraggle attack, DNS, etc. Smurf attacks are carried out using ICMP packets, while raggle uses UDP.

Domain Name System (DNS) is a service that maps a website URL (human -readable) into an IP address (numeric) that will connect the host with the web server. After typing the URL of the website into a navigation tab, the host requests the corresponding IP using a DNS query to the DNS server. This server consults its table, in case a match responds with the requested IP to the client. In case the IP is not found, the DNS server requests in a repetitive way to the DNS root server. In case of a negative, it will provide information about other DNS servers that could obtain the IP request. This procedure is repeated until obtaining the request IP. As with any technology, some vulnerabilities can be explored by attackers such as exploitation of resources of a DNS resolver. fig. 3.9 exemplifies a DNS amplification attack where the attacker uses an army of bots with the spoofed IP address corresponding to the victim and send a DNS request to the different DNS servers. The victim receives an extensive response with DNS replies able to shut down its communication with the external world. This attack is a legacy of traditional networks that still occurs in SDN.

- **Northbound interface threats** Most SDN functionalities can be implemented via applications and these commands are sent directly to the controller with the intermediation of the Northbound API. As standards have not been defined for this interface so far, any vulnerability can be an open door to be explored by malicious users to take possession of the controller. Therefore, the utilization of applications from different third parties could be also a point to be attentive. Not only in terms of compatibility, but also in terms of authentication and resource access to controllers can bring impact to the whole network. The softwarization of networks has brought impressive advantages such as the rapid response to threats, quick deployment of solutions etc. However, SDN inherits the common security issues of information systems such

as authorization, access control, malicious code, malware, etc [73, 14].

Due to all the reasons cited, the security of NBI is crucial to secure the controller. As a consequence the whole network that's the reason that these aspects must be considered in this survey. In the next section, security vulnerabilities regarding NBI will be discussed.

Access control Access control is implemented by security policies that specify the personnel who have a specific grant to a specific system's resource and the type of permission. Any deficient or ambiguous policy, misconfigurations, or bugs could be transformed into an access control failure, not necessarily a failure in the cryptography mechanisms [74]. This assumption is still valid for SDN networks, once the attacker has wrong access to a controller module he could be able to manipulate the network for his benefit.

As an example, suppose an application has correct authorization to access a controller as a traffic monitoring application and it is given READ permission to flow statistics and by mistake is given WRITE permission to flow-rule enabling modifying it on run-time. The user with possession of this information can misuse the data, either unintentionally or deliberately, changing the network environment [50]. This hypothetical situation illustrates that not only under an attack, but misconfigurations could open a door for more serious issues that can have a huge impact on the whole system.

Authentication The lack of standardization of Northbound API and use of third-party applications from different organizations from the controllers can pose threats of wrong authentication or authorization to illegitimate parties providing privileged access to the core services of an SDN network. In other terms, faulty or lack authentication mechanisms lead to improper authentication, as a consequence authorization to modules that must be isolated to the user [74].

In contrast with the Southbound interface with the support of the optional TLS protocol between the control and the data plane, Northbound lacks a standardized authentication mechanism [31]. As a result, literature has been produced to cover this gap but there are still pending studies towards this subject.

Malicious code and flow injection Flow rules are the essential keys to inter-switch communication in SDN networks. OpenFlow switches consist of flow tables and data to impose an action to be endorsed upon the flux arrival. The concept of flow tables are very similar to the traditional networks concept of routing tables. In contrast with routing tables, forwarding tables are modified via Flow-Mod, a message sent from the controller to update, alter and add flow rules in the flow table. Differently, the computation of the routes and modifications of the forwarding path inter-switch is processed in the controller instead of an internal operating system of a network device [14].

For the reasons mentioned in the previous section of application vulnerabilities, a malicious application can manipulate the controller, modify flow actions, and completely mislead communication. For instance, a specific flow rule can be altered instead of forwarding, dropping a flow regarding a specific IP, or being rerouted to another destination and other actions.

Freedom of deployment of network applications by the utilization of APIs offered by SDN permits different entities to provide diverse solutions with less hardware complexity. However, the customization comes with the cost of exposure to bugs, malicious system calls, and other failures.

- **Topology attacks**

  Topology information is an essential component for understanding the position of the devices and the management of SDN networks. This information is processed in the controller and it is crucial to present an abstract view of the network that will be available to the network application. Consequently, any alteration of the traffic will be presented to the network application with a wrong visualization of the network infrastructure. Controllers obtain information from diverse entities to create a big picture of the network device's location such as identification of switches, link distance between them, number of hosts, number of ports, MAC address, etc. Host information is obtained from a module called Host Tracking system (HTS) which is focused on updating the host profile table with the correct identification of a host in the network and its exact position in the network. Inter-switch distance is discovered by OpenFlow Discovery Protocol (OFDP) uses Link Layer Discovery Protocol (LLDP) to create a database of the distances between switches[10].

  Even though all the systems are working together to ensure a clear view of the network topology to higher layers, there is still a lack of authentication mechanisms to avoid any kind of interception by an attacker. As a consequence, an adversarial can easily fabricate fake LLDP links and Packet-in messages with forged information without preventive check from the controller. Among the existing Topology attacks, this work will focus on two specific attacks: host location hijacking and topology poisoning attack. The first one a host impersonates another host to trick the controller into updating the host profile table with the wrong position. The second one has the same idea as lack of authentication, fake LLDP packets are created to trick the controller and update the topology database with wrong information.

  Host Location Hijacking Host Tracking system (HTS) is an SDN module that receives as an input the Packet-In messages and updates the host profile table which is a table mapping all hosts and their respective switch ports attached. This table consists of the following parameters: MAC address, IP address, DPID number (DataPath number), and port number. Every time a Packet-In event occurs, it is performed a search into the host tables to verify if the entry is new or is already existing to apply one of two events: JOIN or MOVE. JOIN occurs when there is no entry in the database corresponding to the host and it is assumed that the host is recently added to the network. Otherwise, during a MOVE event, an entry for the same host is found but with a different location within the network, it is assumed that the host changed its position so this information is added to the host profile table [44] Entries of the host profile event are released when the host leaves the network. The biggest drawback of this module is the complete lack of authentication when a MOVE event occurs. An attacker can forge the Packet-In with wrong position information or use the location from other hosts that are part of the SDN network to fool the controller. The consequences of such an attack can be extended to fake rule creation, flood attacks, resource consumption, initiating other types of attacks such as Ddos etc.

  Another mischievous scenario would be an event where the attacker sends Packet-In with different location information fields to trigger a MOVE event, a huge host of programmed migration. The extra processing of Packet-In may degrade network performance and overload the controller bandwidth resulting in disastrous consequences.

  Link Discovery The Link Discovery process starts when a controller manages to initiate a transmission with a switch by exchanging Feature-Request and Feature_Reply messages. The first message begins with the controller requesting the Datapath ID, ports, etc. that are promptly responded with the Feature-Reply containing the DPID and list of active ports. Provided that information, controllers send an LLDP packet encapsulated in a Packet-out message for each

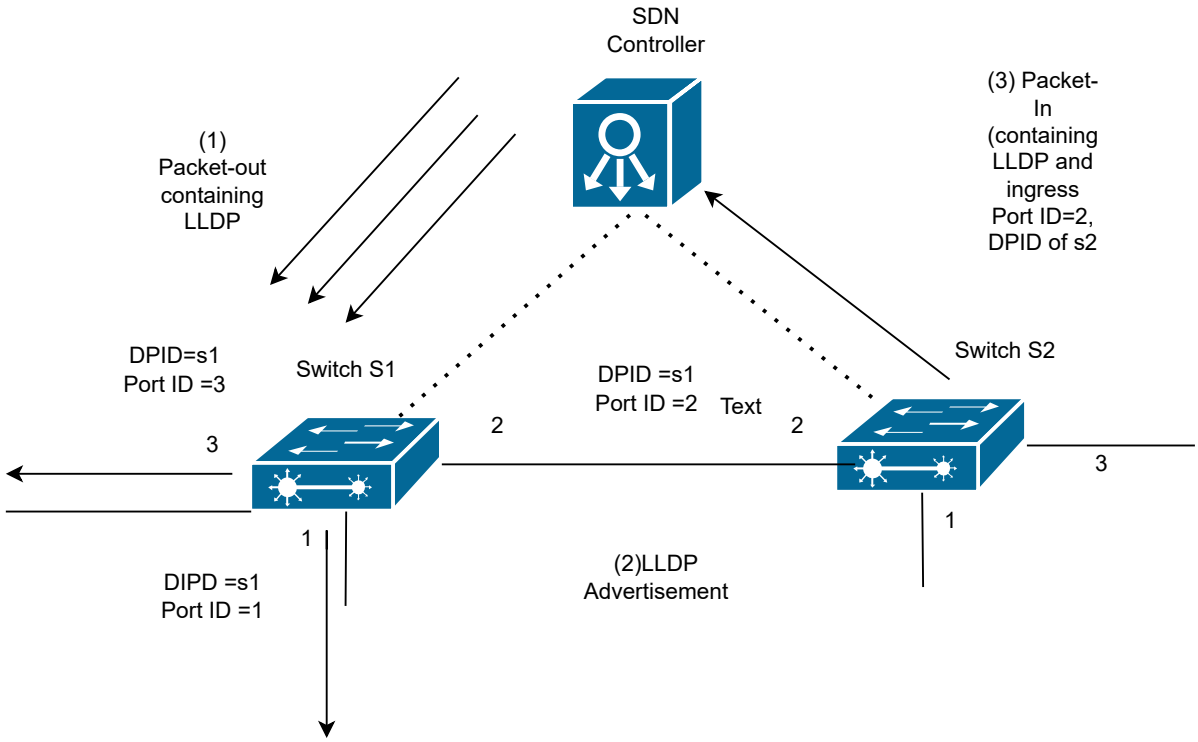


Figure 3.10: LLDP basic functioning

Source: Wang et al. [56]

port of the requesting switch. After receiving the Packet-out messages, the neighbor switches react with Packet-in messages according to the flow rule that states that any Packet-out message must be responded to with an associated Packet-In [10]. These Packet-in messages are sent containing the Data Path of the corresponding switch and the ingress port based on where the Packet-out message was received. Based on these Packet-In messages, a database containing the distance between the switches is created called Management Information Base (MIB) regarding the placement of each switch in the network.

fig. 3.10 exemplifies the process with the support of a simple topology composed of one controller and two switches. In step (1), the controller sends three Packet-out messages, each of them corresponding to the three ports of the requesting switch 1. In the next step occurs the LLDP advertisement in which the packets are sent out through the switch interfaces reaching their neighbors (2). Upon receiving the Packet-out messages from neighbor switch-ports, switch 2 creates a LLDP message encapsulated into Packet-In to be transmitted to the controller (3). This message contains the DPID of the responding switch and the ingress port from where the Packet-out was received. In the case of the figure, switch 2 received the Packet-out from switch 1 in port 2, so the DPIP field was filled with the ID of switch 2 and port number 2 from switch 1 [75].

The Link Discovery process is not free of potential security breaches and also presents the same core issues of the Host Tracking system, issues with authentication of the packet. Consequently, any fabricated field LLDP packet can be considered legitimate and be part of the MIB table Nguyen and Yoo [75] simulated two attack scenarios, one of them is the LLDP fabrication packet that is modified by an external entity and LLDP replay, where the LLDP packet is legitimate but moved from one switch to another.

- **Other attacks**

Fishing It is a social engineering attack in which users are tricked to believe that corrupted sources are trustworthy. A scammer or hacker pretends to be a person or an organization to steal information from the users. Normally, it is fraud applied through email transactions or also SMS messages to steal confidential information such as passwords, Ids, credit cards numbers, and installing malware. And it is considered one of the most common frauds nowadays. Masoud et al. [51] use Kali Linux to create clone websites to imitate websites such as Facebook, Yahoo, etc. Indeed, it was tested hosting in a local IIS webserver and online free hosting.

Personal Hijacking Persona Hijacking is a new type of attack part of binding attacks that target the association of identifiers from different layers in the network. Examples of identifiers are MAC addresses, IP addresses, VLAN ID, etc...In summary, binding attacks target this association of IDs such as ARP protocols, and DNS due to the insecurity of these protocols [76].

Persona Hijacking's objective is to take down the IP address for a certain time until a DHCP release is renewed. The first step of the implementation is to trick the DHCP server into the victim's IP has been released, forcing a reaction of sending back as an offer to the victim's IP. Before sending this offer, the DHCP server sends ARP requests to verify which IP addresses are still in use. The second phase starts when the attacker tries to block the victim from responding to the DHCP server with an ARP Reply. To achieve its goal, an attacker sends fake packets with the IP origin as the DHCP's server and the destination as the victim to the SDN controller. Consequently, the SDN controller creates a flow rule and propagates it to the network devices. If this action happens before the arrival of the ARP request from the original DHCP server, the rule is created considering that the DHCP server has migrated to the actual attacker's position and it is propagated with this information. Due to this lack of synchronization, the victim replies to the attacker instead of the DHCP server with ARP Reply. By that time, the attacker ends DHCP discovery confirming it is the owner of the victim's IP [50].

3.2.2 Indirect attacks

- **Fingerprinting**

  In security, fingerprinting is an attack where the external entity studies the target to obtain knowledge of its characteristics and functionalities to infer its profile. For instance, Shin and Gu [77] used as an advantage of the characteristics of an SDN higher response time of new-flow packets in comparison to existing ones. This information can be used to understand if the corresponding flow is recently created.

  In another example, Shin and Gu [77] used intrinsic characteristics such as hard time and idle timeouts, CPU processing speed and sniffing LLDP packets, and ARP Request as features to detect the type of controller.

- **Southbound interface threats**

Authentication Authentication between entities at the Southbound interface presents risks and this fact is strongly related to the adoption of TLS/HTTP in OpenFlow. According to Opening Networking Foundation, the institution responsible for the adoption of SDN, the adherence to Transport Layer Security (TLS) and Secure Sockets Layer (SSL) is recommended as optional.

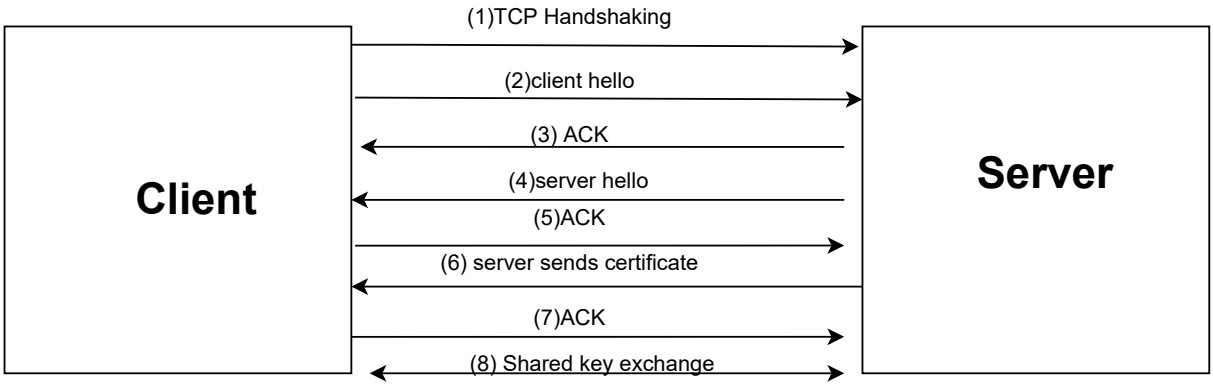


Figure 3.11: Server/client TLS handshaking

Source: Midha and Triptahi [52]

Hence, the vendors are completely free to decide about the adoption according to their business, putting at risk the safety of communication. The hole in authentication creates opportunities for diverse types of attacks, among them Man-in-the middle attacks.

The scheme below the basic authentication process of a TLS authentication for a generic client-server architecture that can be extended to OpenFlow protocol, as it is shown in fig. 3.11. The first step is the three-way handshaking between client and server

1. packet " client hello" is sent to the server along with cipher information
2. A confirmation of the "client hello" is sent back with an ACK packet from the server
3. Server confirms with a packet "server hello"
4. Client confirms "server hello" with an ACK
5. Server sends digital certificate (public key)
6. Client responds with an ACK
7. Shared key exchange
8. Encrypted handshaking messages

The TLS connection initiates after step 2 with the message "client" hello with three parameters: max version of TLS, list of ciphers suites, and a random number. In sequence, the server sends the message "server hello" containing the highest version of TLS supported, and a random number. In step 7, the server sends its certificate to be authenticated and to obtain the public key from the client. Finally, the client generates a key to be encrypted using the public key from the server's certificate in step 8. To finalize the step, both parties have a shared key [78].

MAN-IN-THE-MIDDLE This attack is characterized by an external entity positioned in a communication channel between two end nodes, intercepting the communication (eavesdropping) in an anonymous mode. MitM can be positioned as a consequence of other security issues such as ARP spoofing discussed at the beginning of this chapter. After poisoning its ARP cache table, the sender starts exchanging messages with the attacker host believing it's the destination host. Methods to implement Man-in-the-middle includes session hijacking, DNS spoofing, port mirroring, etc Z. Shu [9].

Kali-Linux is used to perform MitM in Sebbar et al. [54] along with Wireshark for traffic sniffing and Ettercap for ARP spoofing. It mentioned available tools for detection such as Snort, ArpAlert and ArpwatchNG.

In the previous section about Southbound authentication, it was mentioned the adoption of TLS/SSL as an option to secure sessions. However, the protocol is open to impersonation attacks and forged certificates [78].

Data plane saturation The idea of exhausting resources was already touched in the previous sections about Denial of Service targeting the control plane. Different from the previous section, delay or disruption of the controller services are considered a consequence of abuse of the resources of lower layers.

An attacker can use the functionality of table-miss of the flow table, flooding the switch capacity with spoofed messages with random destination IP. If these IPs are not in the range of the network, not corresponding to a flow rule, a Packet-in event is triggered to the controller. A large number of packets saturate the input buffer storage of the switch, later overloading the controller capacity and the result may be large delays or service unavailability if it is not detected on time. The scalability of controller bandwidth has already been discussed and investigated and pointed out as a weakness to be solved[12].

The adoption of a proactive flow rule differs from the reactive flow rule insertion, with respect to the fact that is not generated by an event. Instead, it is previously inserted in the control plane and it can be an alternative to the reactive flow rule insertion. However, there are not sufficient studies to prove it's efficient regarding flooding attacks. Furthermore, attacks are performed by manipulating IP addresses that are out of the range of the network so the attack is mounted regardless of the use of the spoofed IP [56].

- **Reconnaissance attacks**

  According to the Merriam-Webster dictionary, reconnaissance is defined as "a preliminary survey to gain information, or specifically, an exploratory military survey of enemy territory" [79]. Analogous to a military exercise, an attacker would observe the target's weaknesses, strengths, and resources in order to build an action plan to bypass a security defense system. In cybersecurity, an attacker would gather as much as possible information using methods such as eavesdropping, port scanning, and sending probe packets.

  These probe packets are aimed to obtain information such as the number of active hosts, operational systems version and the port number to obtain the type of service offered. Probes can be sent in active mode to the user or employing inter mediator or passively using methods such as eavesdropping, sniffing, and any technique not involved in the establishment of a communication [80].

  This attack aims to obtain vulnerabilities in the system by finding open doors. Packets are sent to each door of the target victim, and depending on the response, it's inferred if the door is vulnerable. The information obtained could be a list of active and inactive ports, services and types, and version of the Operational Systems. The existing types of port scanning are described below [81]:

  SYN scan This technique uses a "half-opened connection", in which the scanner sends a packet with the SYN bit flag. In case the server's port is opened, it will respond with an SYN-ACK. Consequently, the scanner responds with an RTS packet to close the connection.

  TCP scan The scanner tries three-way handshaking with the port to verify if it is opened. In case of an error during the message exchange, an error is returned.

ACK scan This scan technique is performed to verify if the ports are filtered or not filtered by a firewall. Recent firewall techniques block the ACK packet which would facilitate the detection of the presence.

FIN scan This technique can bypass firewall protection. In case the port is open, the server will simply ignore the request. In case the port is closed, the server will send an RST back.

- **Side-channel attacks**
  F.Standaert defines side-channel attacks as "closely related to the existence of physically observable phenomenons caused by the execution of computing tasks in present microelectronic devices "[62] in his work "Introduction to side-channel attacks".This definition is applied to digital electronics. In other words, it is an attack that collects information that is intrinsic to the system such as power consumption, timing and electromagnetic radiation are one of the examples cited by the author. In relation to SDN, J. Sonchack [63] used a timing probe and test stream packets to measure ICMP for a range of IP to obtain a packet pattern and obtain access control lists, routing tables, flow table rules, etc.

3.3 CHAPTER 3 CONCLUSIONS

This chapter presented the attacks and security issues chosen to compose this Thesis and their respective concepts. In the following chapter, it will be introduced the solutions applicable to the concepts highlighted in this chapter.

4

Overview of the security defense and mitigation solutions

This chapter introduces the security solutions available in academia classified according to the previous categorization such as attacks and security issues according to their impact to the controller.

4.1 Solutions addressing direct attacks

4.1.1 DoS and DDoS attacks

DoS and DDoS are considered one of the most lethal attacks able to disrupt an entire connection in an SDN network through the manipulation of hosts to exploit legitimate system resources. Some of the most common attacks such as SYN Flood, ICMP Flood, and HTTP are covered in this section. Furthermore, a special version of HTTP flood called Slowloris is covered. This attack is characterized by an incomplete packet field and low rate that imitate real traffic. Different from the other attacks, DoS and DDoS present an extensive number of articles that make it difficult to cover all the solutions. In terms of numbers, machine learning and entropy-based mechanisms were the most frequent solutions found. Some of them will be briefly discussed in this section.

Machine learning solutions Yang and Zhao [26] designed a framework to detect HTTP flood in SDN in a campus network. Firstly, it is detected as a DDos by the calculation of entropy; it is given a flag "1" and the traffic collection model starts. A sniffer collects the statistical data to be grouped as a feature vector to be the input of the classifier. The classification utilizes SVM (Support Vector Machine) and dataset KDD99. In case a DDos attack is detected, a FLOW-MOD message is sent to filter the packet. The model has shown good accuracy with 0.998. However, other tests and experiments with other machine learning algorithms and datasets would be more precise and realistic.

Statistical-based solutions Tatang et al. [42] proposed a SYN-flood detection mechanism based on the entropy of the IP address. Entropy is calculated based on the comparison of observation samples during a period time with a predefined threshold. In case the computation is lower than

Article	Type of Ddos	Description	Pros	Cons
[26]	HTTP Flood	Framework to detect DDos attack using SVM	High accuracy detection	Tested only one algorithm, only one dataset
[27]	Slowloris	SHDA application detects Slowloris attacks	Low occupation of connection resource	HTTP response time is long to packet not under attack
[28]	DNS Amplification attacks	DNS Amplification detector using flow control mechanism	Bandwidth consumption is stable under attack	No authentication mechanism of flow packets received,
[29]	SYN Flood	SYN-Flood detection mechanism based on entropy	High precision detection, low attack detection time	Hypothetical dataset (hping3)
[30]	ICMP, UDP, TCP flooding	DDos detection mechanism based on joint-entropy	Low false positives, high rate detection	None

Table 4.1: Efforts to mitigate DDos

the threshold is considered an attack, otherwise it is not attack. The test carried out a benign and malicious packet and it was observed in different windows sizes. During the performance tests, the parameter window size impacted the CPU processing, the detection time and the threshold.

Mao et al. [30] designed a flooding detection mechanism based on join-entropy and introduced a new parameter called "flow duration" (time difference between the first and last packet of a flow). The author opted for carrying out experiments to compare the results of single-entropy and joint-entropy with the following parameters: flow duration, packet length and destination port. TCP, ICMP and UDP traffic were generated and the calculation of entropy was based on a selected parameter. Overall, the test results were superior to joint-entropy in comparison to single-entropy.

Application solution Hong et al. [27] proposed SHDA (Slow HTTP DoS and DDoSDefense Application) to detect if an incomplete HTTP request packet is an attack or a legitimate user with a slow Internet connection. A webserver request SHDA to evaluate packets that are suspicious of being part of an attack when the number of open connection surpasses a threshold. SHDA performs a time-out check to verify if the incomplete requests remaining incomplete in a predefined time slot exceeds the threshold value and the controller is warned to create a flow rule to block these types of flows. During the simulations, SHDA has been proven efficient to block attacks in a short time. However, it fails to respond when the packets are considered slow or legitimate.

Flow control mechanism Han et al. [28] proposed a DNS amplification attack detector composed of three modules: Malicious Defender, DNS Checker and Flow -Mod installer. The first module drops all DNS packets arriving from the external sources. DNS checker verifies the IP header

Article	Type of permission	Description	Pros	Cons
[31]	Role -Base	Proposes and administrative model and partial order role	Partial role authorization using the least privilege	No simulations or practical results
[32]	Role -base	Extends the least privilege principle to the relation application and session	Presents inter-session interactions and the low overhead during performance test	False positives, unauthorized operation were allowed to be performed
[33]	Behavior -base	Update permissions dynamically based on network metrics	IDS detection low overhead	Not enough data to be classified as malicious, concurrency, false positives permissions, controller overhead
[34]	Parameterized -base	Enhance granularity of SDN-RBAC introducing the concept of parametrization	Isolation of the resources and more specific permission to applications	Higher overhead in comparison to SDN-RBAC

Table 4.2: Solutions to Access control in SDN.

of incoming flows destined to DNS Public server. Flow-Mod installer creates rules to allow communication between the switch and the server without going through the controller. Despite the simplicity of the ideas, the model does not propose an authentication mechanism to certify if the flows are intended for the DNS server.

4.1.2 Application plane attacks and threats

This section aims to categorize some of the security issues and attacks related to the application plane and the Northbound interface according to the following classification: Access control, Authorization, Malicious application and flow rules. Since this work is focused on the control plane, Application plane security issues were included due to their impact on the controller. As a consequence, the impact will influence that whole network.

During the literature review, few articles related to the data plane and Northbound interface layer have been found in comparison to other layers. However, the development of third parties applications, lack of standardization of the Northbound APIs, and proper authentication mechanism are open to challenges not yet resolved with a plethora of possible solutions.

Access control Applications need to be granted to certain permissions to access network resources. table 4.2 shows the categorization of the articles according to the type of permission. Role-based permission is granted to an application according to the role developed. For instance, an application can be given a role of "ADMIN" and automatically inherit all the permissions related to that

role, including permissions that will not be in use. Behavior-base is a dynamic upgrade or downgrade of permissions according to the network status. Parameterized access aims to provide fine-grained permissions to an application. Using the previous example, parameterized values associate the application with a parameter "Department", switch "S1" etc. In other words, an application is permitted to operate in a restricted department and devices, instead of the scope of a whole organization.

Behavior-based SE-Floodlight was designed as a more secure alternative to the controller Floodlight, utilizing a security extension that applies RBAC (role-based access control) to network application access to network resources in the data plane. In [31], it is proposed an administrative model to alter the relationship among applications, roles and operations allowed and eliminate dependencies. Furthermore, partial role hierarchy is proposed to associate a type of operation to a role according to its tasks and in a more granular mode to avoid issues such as over-privileged access, upgrading and downgrading. In article [32], RBAC-SDN applications are extended to the session-level. For testing purposes, an application was developed broken into two tasks separated into sessions and the permissions and roles were assigned in run time. During the testing phase, false positives. In other terms, permission not allowed to a role associated with a session was authorized.

Behavior-based BEAM was developed to assign permissions to network applications based on the evaluation of network conditions in a predefined time slot. After the end of this period time, permissions can be downgraded or upgraded. However, the scope of permission is not defined and the time slot for evaluation of the network metric and this dynamism of evaluation can require high processing capacity from the controller [33].

Parameters-based ParaSDN was designed to introduce the concept of parameterized access model to guarantee granularity of permissions and ensure a more secure network application. In other words, providing more isolation to resources by associating permissions and roles in a more specific mode to attend a network application ensures the principle of least privilege. Different from SDN-RBAC, the permissions to a role are restricted to the target parameters such as topology, flow, application, and organization. During tests, the model showed almost a linear growth between the latency and number of parameters. However, the processing capacity was proven higher than a controller running SDN-RBAC [34].

Authentication and authorization Authentication is a process of verifying if the application's credentials match with the authentication database of the system it intends to access. Normally, it is provided with a "username" and "password". After finishing this process, authorization grants access to predefined resources. table 4.3 shows diverse techniques to assure a secure process. Despite the efforts to solve security issues, little literature has been produced to address authentication ,and authorization Consequently, third-party or malicious applications can benefit from these vulnerabilities and obtain privileged information and access.

Application Authentication system Cui et al. [35] introduced an application authentication system interface that provides access to authorized users and creates a log of unauthorized ones. Firstly, a set of permissions are granted to a network application and then the authentication

Article	Type of authentication	Description	Pros	Cons
[35]	Application authentication system	Propose an application authentication mechanism that audits illegal access	Audition of illegal access	The system functioning is not well explained, heavy CPU processing
[37]	Token	Propose a Trust framework comprising an authentication, authorization and trust module	OAuth 2.0 as secure protocol	Tests are not conclusive, no practical results, design phase
[38]	Token	Design a NBI utilizing token-based authentication	Lightweight performance during an authentication	It is on design phase, no practical results
[36]	Digital signature	Proposes a controller independent plug-in to fulfill the gaps of application security DN-RBAC introducing the concept of parametrization	Investigate the security vulnerabilities of five commercial	No practical test, feasibility study
[39]	IEEE 801.x	Security module based on IEEE 801.X and EAP authentication based on MAC layer	Security module based on IEEE 801.X and EAP authentication based on MAC layer	None

Table 4.3: Authentication and authorization solutions in SDN

is performed by an application authentication system. In this system, an unauthenticated application is redirected to the login page to verify its certificate status. In case the certificate is valid, permissions previously granted will be verified. In case the application is not certified, the system will refuse to provide access and perform logging of unauthorized access as malicious. The article fails to explain the basic functionality of the authentication system presented and the performance tests resulted in high overhead in comparison to the application without authentication process as expected.

Authentication token Aliyu et al. [37] proposed a trust framework divided into three modules: Authentication, Authorization, and Trust. In the authentication phase, a procedure evaluates the correctness of a network application and token generated and exchanged at the first contact. The authors decided to test the procedure with a firewall application whether is consistent with the application and the token generated. Oktian et al. [38] designed a more secure Northbound interface utilizing token authentication and authorization. The article presents two authentication schemes: direct and delegated. During the direct authentication, the application exchanges its keys for an access token to communicate with the Authentication Server. During the delegate, firstly the access token sent will identify the user and then the application to obtain network resources.

Digital signature ControllerSEPA [36] was developed as a REST-based controller-independent API plug-in to address security issues such as flow rules conflicts, malicious application access, etc. In terms of authentication, ControllerSEPA implements mutual certificate-based authentication using TLS protocol with a cipher suite using Diffie Hellman with an ephemeral key following the guidelines of the article [82]. The author opted for a feasibility study of ControllerSEPA concerning the controllers available in the market. In terms of authentication, it is mentioned the effort to use digital certificates for authentication and token to authorization but no practical result or test were implemented in this niche.

IEEE 801.X AuthFlow [39] was a mechanism based on IEEE 801.X standard and EAP protocol to provide authentication above the MAC layer. A client starts a connection with the controller which passes this request to an entity called authenticator. The authentication replies to the host requesting its identity which will be sent to the Authentication server (Radius) that confirms the identity of the host. EAP is the protocol running from this path from the client until the authenticator that translates to the format understandable by RADIUS server.

MALICIOUS APPLICATION THREATS AND FLOW INJECTION The deployment of network applications results in a myriad of harms to the controller. For example, an attacker can reuse a legitimate API to disable a firewall as an opportunity to launch an attack. table 4.4 presents some detection solutions to discover beforehand potential threats such as the analysis of system calls and API that performs critical operations.

Behavior profile solutions Lee et al. [40] presented a framework called Indago able to build a data structure named Security-Sensitive Behavior Graph (SSBG). The features extracted from this graph were used to discriminate an SDN application as malign or benign. Security-sensitive

Article	Applied solution	Description	Pros	Cons
[40]	Behavior profile	Framework based on SSBG to classify an SDN application as malign or benign	Clustering technique presents low false positives, high detection accuracy	Fails to detect malign application, small dataset
[41]	Machine learning	Framework based on machine learning to monitor system calls utilization framework comprising an authentication, authorization and trust module	Random Forest presents a high F-1 score	It is not explained the origin of the malicious application, lack of performance results
[42]	SDN-GUARD	Dual view detection mechanism to protect against rootkits	Low detection time, adaptability to controller	No realistic tests with rootkits, dependency of OS, time elapse

Table 4.4: Malicious code detection in SDN

was the term to define the Northbound APIs that could handle critical resources. For the experimental part, the authors collected security sensitive APIs from three popular controllers to create the dataset. During the section on insights, it was mentioned that security sensitive APIs were more likely to be used in malicious applications. One of the good points of this framework was that high accuracy to detection of malign applications. However, it is still necessary to improve the classification task with a larger and diverse dataset in order to decrease the evasion issues.

Machine learning solutions Chasaki and Mansour [41] designed a framework to detect harmful applications from system call feature using machine learning classifiers. The training data was obtained in offline mode from the system call extraction from malicious and benign applications along with real-time data capture in online mode. During the learning process, it experienced the multi model-training using Random Forest, Support Vector Machine (SVM), Nearest Neighbors Classifier and Gaussian Naive Bayes. The article lacks more detailed results of the classification and it does not investigate its impact on the controller.

SDN-GUARD Tatang et al. [42] created a rootkit detector that performs a dual view comparison between components in a network. One view is positioned at the Northbound interface, while the other is positioned between the controller and the switches to scan flow-mods. The Flow-Mod messages are extracted from the switches and it is compared with another view. If the information is not the same, a new Flow-mode is generated to return the original status. Basically, the idea of SDN-GUARD is not only to detect rootkits, but establish the status of the network at the beginning.

Article	Spoofing type	Description	Pros	Cons
[8]	ARP	an ARP spoofing detection and mitigation mechanism	Interoperability with legacy networks	Not clear what would be the existing technique to be compared
[24]	ARP	an anti ARP spoofing mechanism that overwrites ARP packet fields	Low overhead, low RTT	Can't discriminate two ARP Replies with the same source hardware source field
[25]	IP	Firewall application anti-sniffing and anti-spoofing	Rules are efficient to allow/deny	Lack of conclusive experiments to prove anti-spoofing

Table 4.5: Spoofing solutions in SDN

4.1.3 Spoofing attacks

Spoofing is characterized by the falsification of a source to be considered legitimate. It can be used to steal personal information, spread viruses, mislead network topologies, or to reroute traffic among other threats. Normally, it is used to introduce other attacks such as DDos, Man-in-the-middle, and others. Despite being a legacy attack, spoofing is a recurrent threat in SDN networks to initiate the attacks mentioned below. Regarding SDN, few articles have been found mostly related to IP and ARP spoofing as it is cited in table 4.5.

Detection and mitigation ARP spoofing technique Ubaid et al. [8] proposed a system to detect and mitigate ARP spoofing attacks in Hybrid SDN networks. Topology consists of legacy and SDN switches, host, a server containing the detection modules and controller. The detection process starts when an ARP packet is transmitted and then redirected to a SDN switch. In the case of table-miss, the packet is resent to the server to verify the IP and ARP origin. In case it is part of the network, it will be forwarded to the destination. Otherwise, it is dropped. Topology manager that detects the location of user and attacker generating a graph at a certain time slot. The technique has successfully outperformed other existing techniques in terms of CPU utilization, attack detection and other metrics. However, it is not clear what kind of techniques the authors are referencing.

SARP NAT Alharbi et al. [24] designed SARP-NAT, an ARP detection mechanism. When a regular ARP spoofing attack is launched, SARP-NAT creates an entry in a pending ARP Request list (pend-req) and overwrites ARP fields such as Source Protocol Address(SPA) and Source Hardware Address (SHA) with IP and MAC addresses that are not assigned to any host. Then ARP Request is sent to the target host which stores the new field value in its ARP table. Before the ARP Reply procedure, SARP-NAT consults the pend-req list to find an entry. In case there is a match, the values of the fields Target Protocol Address (TPA) and Target Hardware Address (THA) are replaced with original values. One of the biggest drawbacks is that the system can't identify two ARP Replies with the same SHA field.

Article	Type of topology Attack	Description	Pros	Cons
[43]	Host Tracking system	Presents a new type of HTS attack and its countermeasures	Able to detect inactive hosts	High ARP traffic
[44]	Host Tracking system	Host hijacking mitigation technique using machine learning	Low CPU, memory and attack response time	None
[45]	Link Fabrication Attacks	a Topology discovery MitM	a Topology discovery MitM	High CPU and memory consumption due to topology calculation
[46]	Link Fabrication Attacks	an Relay-type Link Fabrication Attack detection using comparison of statistical model between a baseline and a new link	Vetting period is not influenced by network congestion	Trade-off between sample of the vetting period and attack detection, trade-off between traffic congestion and statistic mean
[47]	Link Fabrication Attacks	an ant Relay-type LFA using cryptography mechanism	Blocks Relay-type attacks	Lack of conclusive results, high overhead in comparison

Table 4.6: Type of topology Attack

SAVSH Gautam et al. [25] performed experiments on layer-2 POX and layer-3 Ryu controllers to develop a firewall application for anti-sniffing and anti-spoofing. This application is driven by pre-defined rules and policies which compare the header fields of the incoming packet with the firewall rules. If there's a match action will be performed, otherwise it's dropped. The idea is not original and there have not been carried out enough experiments to prove the effectiveness of the solution against spoofing.

4.1.4 Topology attacks

Topology attacks aim to disrupt the controller services of mapping the end devices and their respective position in the network. Considering the articles surveyed, the articles were divided in table 4.6 according to the following category: Host Tracking attacks and Link Fabrication attacks. Host Tracking attacks are threats towards the Host Tracking system, a database containing the identification of all the hosts and the network (IP and MAC) and position (DataPathID). Link Fabrication Attacks are related to OFDP protocol which uses LLDP packets to discover switch's location in the networking via Packet-out messaging.

Host Tracking Attacks

HTS Free-Port Attacks Baidya and Hewett [43] introduced a new Host Tracking system attack called "HTS Free-port" attack and mitigation. Differently to other forms of attacks, Free-port takes advantage of a vulnerability of OpenFlow protocol during the port-description exchange. This happens when no port update message is sent to the controller when a link fails or the port is free. An attacker can take advantage of the inactivity of the victim to send a crafted Packet-in to mislead the controller about the victim's position. The countermeasure suggested was the transmission of LLDP packets to sense the active ports followed by an ARP Request to force a reply from connected hosts.

SLAMHHA Nagarathna and Shalinie [44] proposed SLAMHHA, a decision tree-based algorithm to mitigate flooding of Packet-In messages when a wrong MOVE event is triggered. The attacker creates a high volume of packets crafted with misleading information regarding location. After the training process, a model is created to classify the traffic according to attack impacted parameters as an attack or legitimate traffic. Illegitimate classified traffic is promptly blacklisted. Experiments showed that SLAMHHA outperformed a cryptography solution previously studied in terms of attack response time, CPU processing and memory consumption.

Link Fabrication Attacks

Correlation-based Anomaly Detection (CTAD) Chou et al. [45] proposed a mechanism to countermeasure the Topology discovery man-in-the-middle attacks called Correlation-based Topology Anomaly Detection (CTAD). When an advent of a Topology Discovery attack occurs, a large volume of LLDP packets are sent to a specific port to be compared with the current topology information. In case the correlation is above 0.7, the entry is deleted. In case the link is new in the topology database, the Time-Difference Analyzer calculates the Round-trip time.

Relay type Link Fabrication detection attack Smyth et al. [46] proposed the first paper to design a mechanism anti relay-type in Link Fabrication Attacks using statistical methods. This article suggests the use of a vetting period for new links in the network. During this period, latency samples are captured and confronted with a statistical model of a benign link called the baseline. One of the drawbacks of this solution is the trade-off of a sufficient number of samples to detect an attack. The more the number of vetting samples, the higher probability of wrong attack detection.

Anti-relay Link Fabrication cryptographic mechanism Alharbi et al. [47] suggested as a countermeasure to link spoofing the usage of cryptographic Message Authentication Code (MAC) to authenticate the LLDP packets. Tests were performed in POX controller using HMAC, a keyed-hash-based message authentication. To combat the Relay-type attack a secret key is randomly generated at each topology discovery cycle for every LLDP packet. To address LLDP authentication, a MD5 hash table at the controller using Chassis ID and Port ID field as identifiers is suggested. The idea is not sustained by experimental results.

Article	Attacks	Description	Pros	Cons
[51]	Fishing	A fishing detector using neural networks and SDN	Good linear regression fit, MSE of 0,08 and low error rate	No performance results shown.
[50]	Persona Hijacking	An experimental test in two scenarios of duration of Personal Hijacking	None	Not considering complex topology

Table 4.7: Other technique solutions

4.1.5 Solutions to attacks classified as Other

This section aims to group the articles that could not be categorized in the previous options. The first chosen Fishing is part of the so-called social engineering attacks in which the attacker claims to be a trustworthy person, enterprise, or organization to steal data from the victims. The attacker sends messages via e-mail, mobile applications, and logins as a "bait" in order "to fish " the victim. The victim enters his/her personal data which is promptly available to the attacker[51]. Persona Hijacking is a type of binding attack in which the attacker impersonates the victim to obtain its IP by exchange of a fake DHCP message. In the meantime, the victim is prevented from reclaiming its IP until another event of DHCP lease[50] .

Fishing attacks M. Masoud et al.[51]proposed a fishing detection algorithm utilizing the properties of SDN networks and neural networks. During the testing phase, features were extracted in offline mode without any SDN implementation. Website cloning was performed utilizing Kali Linux to imitate social network pages such as Facebook,Yahoo, and Hotmail. The neuron weights obtained during the training process were implemented as a Python function installed in the controller. According to the function output, switches were instructed to drop or send packets. Finally, the algorithm was tested with two distinct hosts, Windows, and Linux, in four different scenarios varying the host of Webpages. For all scenarios, the access login to illegitimate pages were promptly denied.

Personal Hijacking S. Anggara et al.[50] tested how long Persona Hijacking attack would last and the time to perform two scenarios, first the DHCP and controller were merged and second where the DHCP is located externally. In the first scenario, DHCP runs as a module of POX controller which does't require IP validation (ARP Request and Reply messages). In the case of the second scenario, it's used UDHCP as DHCP verifies IP liveness by sending ARP messages before IP offer process. During the resulting tests, the attack has taken low values with some difference of 10 seconds.

Article	Techniques used	Description	Pros	Cons
[48]	Time and protocol-based	Techniques to discover policies parameters	None	Some tests are not sufficient to characterize a controller
[49]	Time-based	Exploration of specific metrics to	Presents a new form of attacks	Fluctuating error of estimation of timeouts

Table 4.8: Fingerprinting threats technique

4.2 Solutions addressing Indirect Attacks

4.2.1 Fingerprinting attacks

It is a method to identify a device by a combination of features provided by device configuration and its current state. In this survey, the selected articles presented in common the calculation of time-out values to obtain the network policy endorsed. In order to facilitate the classification, table 4.8 shows the categorization of techniques used as follows: time and protocol-based. While time-based techniques explore the time-out values, protocol-based explore the latency related to the utilization of protocols. For example, the duplicate presence of ARP Request is a characteristic of a specific controller.

Time-based and protocol-based technique Azzouni et al. [48] carried out experiments to discover the controller type according to its characteristics, dividing it into the following categories: Timing Analysis and Packet-Analysis. During the Timing Analysis, a series of experiments are carried out to obtain time-out values to be stored in a timeout database. Moreover, the processing time of packets is calculated using the results from the time-out experience to create another database. Packet-analysis used the time period to retransmit LLDP packets and the packet length to discover controller type. Some tests were not sufficient to have a conclusion about the type of the controller in use. Other features may be further explored to provide a definitive.

Time-based technique Ahmed et al. [49] developed an algorithm to calculate flow time out values related to SDN controllers through the calculation of Round Trip Time (RTT). Similar tests are executed to discover whether matching rules are allowed or blocked by the controller. Furthermore, the authors show techniques to infer when flow entry tables are full. Consequently, major packet drops and an increase in RTT are detected. Moreover, a new attack is presented called Flow Table Entry Attack (FTEA) which the attacker takes advantage of time-out values to infer the maximum capacity of a switch entry flow table.

4.2.2 Reconnaissance and port scanning attacks

Reconnaissance consists of collecting information about services in use, ports and operational systems, detecting the presence of firewalls and identifying vulnerable resources to launch an attack. Once it is gathered the strengths and vulnerabilities of a system, the fabrication of an attack is facilitated. In this section, port scanning was grouped in the same category as reconnaissance, as the latter uses port

Article	Applied technique	Description	Pros	Cons
[59]	Flow rate	a port scanning detector	periodic high CPU consumption	None
[60]	IP validation	a port scanning method using TCP-FIN	Low overhead, no impact at the controller	Port scan process of scenario with 50% loss is not explained
[61]	Firewall placement	anti-reconnaissance solution utilizing distributed firewall	Pre-defined policies adds extra security to mitigate false flows	No performance tests.

Table 4.9: Reconnaissance and port scanning threats technique

scanning at the first stage of information gathering. Port scanning is the process of finding available ports that are active and exchanging data.

Table 4.9 shows some of the articles selected during the surveying process and the categorization was in accordance with the applied technique : flow rating, IP validation and distributed firewalls. In [59], authors focus on starting the port scanning only after the detection of a potential flooding attack, and a high flow rate towards the controller. In [60], it is presented as an alternative solution to discover potential spoof IPs. Lastly, [61] presents a method to prevent reconnaissance using placement of firewalls for each switch and a firewall application assure policies and verify false logins.

Flow rating Ono et al. [59] created a port scan detector based on statistics messages to identify hosts generating a high volume of Packet-Ins. When suspicious behavior is observed, switches are instructed to send statistics messages to the controller. In case of detection, the flow entry is sent to blacklist the host. Comparative studies showed that CPU consumption and traffic are only high in the event of port scanning in comparison to methods with periodic nature such as polling.

PMTS Patil et al. [60] created a model to defend SDN networks against malicious TCP-SYN attacks. The model consists of sending TCP-FIN packets to verify if the IP in use by the attacker is legitimate. When an attack is detected, the port scanner validates the pair source IP and port of the suspicious host using fabricated TCP-FIN packets. In case it is confirmed to be illegitimate, flow is removed from the flow table and blacklisted. Comparative studies with other SYN flooding solutions have shown that PMTS presented a better or equal performance to similar solutions.

Firewall placement Alshamrani [61] proposed to limit the exploration zone of attackers by utilizing a series of distributed firewalls for each switch instead of centralizing the solution to the control plane. Moreover abnormal traffic behavior is monitored thanks to the statistics fields of the flow table. Furthermore, an application SDN firewall is deployed to maintain predefined security policies in the format of flow rules. For every request, the application will consult its policy to find and match to be installed as a flow rule. This firewall application solution creates a log of all

illegitimate login attempts. The test proved that the method could drop malicious connections and the feasibility of predefined security policies.

4.2.3 Data plane and Southbound interface attacks and security issues

This section aims to categorize some of the security issues and attacks related to Southbound interface according to the following classification: Authentication, Man-in-the-middle, Packet injection and saturation attacks.

Similar to Northbound interface threats addressed in the previous section, the presenting threats are targeting specific assets and resources due to their vulnerabilities. Due to SDN centralized design, its repercussions are sensed in the control plane, even when a target is not focused on it. For example, a spoofed IP packet sent at a high rate triggers a table-miss event that generates Packet-in to overload controller bandwidth. Despite the triggered event happening at the data plane, the consequences are directly impacting the controller.

Authentication Southbound interface utilizes TLS (Transport Layer Security) and SSL (Secure Socket Layer) to exchange cryptography messages between clients and servers. The protection is based on public and private keys, the first from a certification authority, while the other is a shared key between network devices and controller. Missing authentication, expired certification or any other issue during the handshaking ensures the channel. As a consequence, it exposures the controller to attacks.

Table 4.10 displays only two articles regarding solutions to enforce the security of the TLS channel. Little literature has been found which concludes there are many efforts going on and opportunities for application of update solutions such as the case of the quantum key [53].

Extended TLS/SSL protocol Midha and Triptahi [52] proposed an extended version of TLS protocol adding extra security during the transmission of "hello" messages ("sender hello" and "client hello"). A sequence of 16-bits is appended to the messages. This sequence is the result of the operation of the Public Key with a randomly generated sequence. After the transmission, the client or server must identify the original sender by retrieving information from a look-up table. The article does not mention how the keys are created and if they are randomly generated.

Quantum key Mahdi and Abdullah [53] introduced a solution to enforce TLS/SSL channel with a hybrid key. The article presents the co-existence of classical and quantum channels and authentication. Both encryption channels work in parallel. In the casE one of the entities not supporting the Quantum Key Distribution protocol (QKD), configuration is promptly rolled back to standard TLS protocol. Even though quantum keys are proven efficient cryptography tools, the article does not demonstrate the concurrency between both technologies.

IBC Encryption J.Lam [83]proposed the use of IBC encryption as an alternative to guarantee the security during the TLS authentication for East-West communication between controllers and the Southbound channel between controller and switches. Focus on the latter, controller acts as the PKG only by generating private keys. Public keys are generated from any identification of the switches or host such as MAC, DPID, etc. During an inter-controller migration, controllers

Article	Solution category	Description	Pros	Cons
[52]	Cryptography	An extended security version of TLS/SSL	None	No concrete results or experimental tests, 16-bit keys generation are not mentioned and no operations are mentioned
[53]	Quantum cryptography	TLS channel security improving utilizing quantum key	Impossible to be broken by Brute-force	No experiments result to prove the hybrid key, CPU demanding
[83]	IBC Encryption	Reinforced encryption to TLS protocol	Protection during switch migration to another control,less processing in comparison to other techniques	Evasion tests are needed to ensure the security
[84]	TLS Handshake	A TLS handshaking verificator	Verification of exchange information during each step	Latency to validate the handshake process
[85]	SECA	A Key generator application.	The usage of certificate type and IP to prevent password stealing	Prone to human configuration error and no measure to prevent certificate forgery

Table 4.10: Authentication threats technique

will smoothly process the handover by providing private keys to the candidate switch.After the handover is finished, the old private key provided by the previous controller is discharged.

TLS Handshake verification A. Ranjbar [84]introduced a framework to enhance TLS security based on SDN architecture. A police-box module is added to the controller to verify plain-text handshake messages parameters such as validity of certificates, date of issue, etc. When a client and server attempts to establish a communication, OpenFlow switches intervene during the handshake process and route the messages to the controller. There policy-box crosses the parameters of the message with the local policies,in case the verification is successful the SDN controller contacts directly to the intended destination(either the client or server). This process is repeated for every singles message during the handshake process until the installation of a FLOW-MOD message is sent to the intermediate switches between the client and server to allow the communication. One of the weakest points of this approach is the latency introduced before initiating a secure communication. Regarding that the process is repeated for every single step during the handshake, the presence of concurrency connection can potentially overload the controller and the link bandwidth.

SECAS B. Yigit and Alagöz [85] proposed SECAS for ensuring TLS security via a key generation application. The application is under one CA and parameters exchanged during the handshake process are configured by the system administrator. The process of the application starts creating a key pair that issues a certificate according to the parameters chosen by the administrator. The storage of the key pair and certificate are protected by password following the security policies deployed. The IT staff retrieves the specific certificate for each SDN entity via user-interface and the key pairs according to parameters previously input and the network policy. Moreover, the IT staff can deny the certificate that is no longer trusted and blacklist it. One of the biggest drawback presented is the fact that the process of obtaining the key pairs and certificate is manually performed.

Man-in-the-Middle (MitM) This attack objective is to place an attacker between two points, in an anonymous transparent mode to eavesdrop on the real communication between client and server. The interception can result in DDos, tampering, etc. Eventually an attack can take possession of the controller and execute malicious action with low chances of being detected.

Table 4.11 shows the few solutions available in the academic databases to address these problems. However, the material found was not sufficient and feasible. This lack of material can be also seen as a potential for new academic works and solutions to be presented in the future.

MitM detection Sebbar et al. [54] reproduced an MitM attack scenario using security commercial softwares and tools. an ARP spoofing is initiated to redirect the traffic by Kali Linux tool. During the simulations, the sniffer takes advantage of the fact that the ODL controller is not secured by HTTPS protocol to proceed with the attacks.Other commercial tools are mentioned but no tests were performed such as Ettercap (sniffing and packet capture) and SSLstrip (intercept HTTPS traffic).

Article	Solution category	Description	Pros	Cons
[54]	Detection	Reproduces a MitM attack using ODL controller	None	No concrete results of the experiment, no performance tests
[55]	Mitigation	proposed a MitM mitigation mechanism comparing fields with a datastore	None	Unrealistic CPU processing time, No experimental results proving the efficiency of the algorithm
[86]	Mitigation and detection	Proposed a MitM detector using DWT	None	Unrealistic CPU processing time, No experimental results proving the efficiency of the algorithm

Table 4.11: MitM Threats Techniques

MitM Mitigation N et al. [55] proposed an algorithm to mitigate MitM attacks in SDN. When an ARP incoming packet arrives at the controller, IP and MAC fields are compared with the information stored in the called mac-to-ip table. In case of a mismatch, the attack is detected and the packet is dropped. The experimental results show that the algorithm presents a lower CPU utilization in comparison with the moment before the attack for both POX and NOX controllers and low execution time. No results were shown proving the efficiency of the mitigation besides a decrease of CPU processing during an attack is unrealistic.

DWT MitM detector and classification J. D'Orsaneo [86]proposed a method to identify traffic abnormalities typical of SDN MitM using DWT. Information such as number of bytes, packets and delayed data are collected and represented as a sampled signal. This signal is processed by the DWT to obtain the coefficient of high level and low level frequencies. Then, these coefficients are reconstructed to be compared with a predefined threshold. The detection phase is performed after the simulation of a MitM attack, a DDoS and a congested network. For classification was observed if the traffic presented no abnormalities in terms of number of bytes and packets and a suspicious traffic pattern of delay data. Otherwise, the traffic was classified as DDoS or just a congestion.

Packet injection Malicious manipulation of packets in SDN can lead to disastrous effects such as misleading the Topology Management, disruption of controller and infrastructure's resources and applications. Considering these risks, articles reviews were performed to categorize potential solutions to the packet injection attack. In this kind of attacker, the hacker forges packets with random field values to force a table-miss, consequently, triggering Packet-In messages. These forged packets trick the Topology Management system and overload the controller processing and bandwidth.

Table 4.12 categorizes the articles according to the solutions to the attack : detector or authenticator.

Article	Solution category	Description	Pros	Cons
[22]	Packet-injection detector	a defense mechanism anti packet injection attacks	Checks vulnerability of port status message	Detection is slightly inferior to other solutions, accept forging MAC address
[87]	Packet-injection authenticator	A Defense and mitigation mechanism anti-packet injection attacks	Legitimates Packet-in messages	high bandwidth consumption
[88]	Packet-injection authenticator	a hardware defense mechanism anti packet injection attacks	Outperformed PacketCheck [22] in all performance metrics	None

Table 4.12: Packet injection Techniques

The first definition is self-explanatory and a mechanism recognizes the imminence of an attack. Packet-in authentication is a solution to defend the controller against spoofed packets. Even though only three solutions were selected, and all of the three presented feasibility and comparative studies to certify the results.

PIEDefender Li et al. [22] proposed a mechanism to detect packet injection exploitation attacks before opening space for flooding attacks. The mechanism confronts the Packet-In messages triggered during an event of migration with the controller host profile. Furthermore, it assures that OpenFlow port status vulnerability is not exploited. During the Ddos detection, features are input to a supervised learning SVM to classify as an attack. Comparative studies have been carried out to certify the detection capacity of PIEDefender during an injection attack. The slightly lower capacity is related to the adoption of MAC Address forging.

Packet-in authenticator Deng et al. [87] designed a detection and mitigation mechanism to combat packet injection attacks called PacketCheck. When the controller receives a Packet-in message, MAC address is extracted and confronted with an internal database (host table profile) containing the list of all hosts with the following parameters: DPID (Switch ID), Inport (switch's interface connected to the host) and host's MAC address. In case the entry is found but the other parameters are not matched, the packet is considered illegitimate and instructions are given to be dropped. Alshra'a and Seitz [88] designed a hardware detection mechanism against injection attacks called INSPECTOR. Inspector authenticates Packet-in messages retrieving information such DPIP (Data Path Identifier of switch), In-port number, MAC address and IP address. In case the Packet-in passed the verification, it is allowed to be sent to the controller. During the comparison tests, INSPECTOR was superior to PacketCheck [22] in relation to all performance metrics.

Article	Solutions	Description	Pros	Cons
[56]	FloodGuard	Ta mechanism to mitigate saturation attack using proactive rules	Avoid dropping legitimate packets during an attack	None
[57]	Flowsec	a detection mechanism anti-saturation attacks using rate limiting	None	No feasibility studies related
[58]	Lineswitch	an detection and mitigation mechanism using proxying	Strong resilience to SYN-flood, only TCP handshaking packets	None

Table 4.13: Switch saturation Techniques

SATURATION ATTACKS Saturation attacks are a type of Denial of Service positioned between the control and data plane. It is triggered by the transmission of multiple Packet-in to exploit controller's bandwidth and CPU processing. During the process of research, Floodguard [56] defines the concept of proactive rule as an alternative to the current reactive flow rule. Proactive rules are computed by the controller and transmit to the switches.

FloodGuard Wang et al. [56] proposed a proactive rule analyzer and packet migration to combat saturation attacks at reactive controllers. In case of a flood attack, packets are temporarily stored at a module called data plane cache in a separated buffers queue until being released to the intermediate module Mitigation module. Packets are converted into a Packet-in and sent to the controller with a rate limit. The usage of proactive rules is independent of an attack event and dynamically installs forward rules according to network conditions without passing by table-miss event.

Flowsec Tian et al. [57] proposed a mitigation algorithm called Flowsec. The algorithm compares the traffic with a predefined threshold. If the values are above, traffic is halved. Results are not clear and the basic functioning.

Lineswitch Ambrosin et al. [58] proposed LineSwitch, a mechanism to detect and mitigate SYN-Flooding attacks target the controller. LineSwitch proposes an improvement in relation to other mechanisms, forcing the attacker to continue the communication only if SYN packets are complete during a handshake process. In case the packets are incompleted, the system would count the suspicious IP and blacklist it. In case, the attacker sends a repetition of the SYN packets, it would be proxied with small probability to be considered as an attack. During the attack simulations, Lineswicth presented an attack detection with low overhead in comparison to its counterparts.

4.3 SIDE-CHANNEL ATTACKS

These attacks are characterized by the exploitation of the functionalities of a system rather than vulnerabilities. In [62] , the author provided examples of information to be obtained by attackers RF

Article	Solutions	Description	Pros	Cons
[62]	Randomized response time	A technique utilizing randomization to fake probing packets delay	Low overhead (probe packet pipeline is parallel to other packets)	None
[63]	Timeout proxy	Utilization of threshold techniques to cheat detector into visualizing controller's overload	It does not interfere with OpenFlow or other parts	None
[89]	Obfuscation flow	A countermeasure to KEY attack faking identification of flows	It uses properties of OpenFlow protocol to cheat the attacker	None

Table 4.14: Side channel attacks solutions

transmission, cached information, and power monitoring. The Table 4.14 shows a few contributions found during the literature research regarding the topic. The categorization selected was based on the defense approach utilized in each article.

Randomized response time F. Shoaib et al.[62] proposed Netkasi which is a solution to use randomization techniques to mask the time information from probing packets sent by an attacker. In the first stages, probing packets are filtered and have their response time parameters calculated such as maximum, minimum, and average . A random number is selected between the upper and lower boundaries calculated in the previous step. This random number becomes the delay to be associated with the packet. The test results have proven the feasibility of the technique as an incremental delay was visible in comparison to probing packets that reached the controller.

Time-out proxy J. Sonchack et al. [63] presented an attack system utilizing time probing packets along with test stream packets based on statistical techniques to obtain information to learn network characteristics. A defender mechanism was installed between the controller and the switch in order to not allow attackers to obtain the controller timing processing to foresee an opportunity to launch a DDoS attack. The idea is to constrain the controller processing time after flow arrival using a threshold. The threshold is set in an optimal way the attack does not perceive a potential saturation attack. Suggestions in terms of control plane defense are to install a log to register to obtain the number of hosts performing timing attacks.

Obfuscation path M. Conti et al. [89] introduced a flow-table side attack called in which the attacker obtains the flow table information of a switch without being detected. How the attack is carried out is not implicit in the article. Basically, the attacker sends probing packets to force a flow rule installation. By the cross information between the probing packet output and the flow installed at the switch, the attacker can deduct the network policies for that flow rule along with security defense strategies such as threshold. With this information, the attacker can evade the defense mechanism implemented at the controller. As a countermeasure was offered the flow

obfuscation deals with the property of the OpenFlow protocol that flows can be modified at each switch forwarding . Basically, the idea is not allowing the attacker to recognize the flows as its own in order to obtain the applied network policy

4.4 CHAPTER 4 CONCLUSIONS

This chapter presented the attacks and security threats already depicted in the previous chapters with their respective potential solutions found in the literature. The next chapter will closes this Thesis with a discussion regarding the solutions and future improvements.

5 Discussions and future works

This chapter proposes a discussion regarding the articles selected to broaden the perspective of the potentially solutions available and the open possibilities.

5.1 Discussion

This section presents some points highlighted during the surveying process that can potential reflect the next research directions.

5.1.1 SDN vulnerabilities

The logically centralized controller manages the whole network and concentrates diverse functionalities such as resources, abstraction, and provisioning of a global view of the network. Hence, it is efficient to detect abnormal traffic behavior on time. Despite these advantages, the accumulation of functions places the controller as a single point of failure, a failure or error is capable of disrupting the whole network. Consequently, DoS and DDoS spotting the SDN control plane is considered lethal to the correct operation of the network. These attacks aim to flood the network with a high volume of requests and high throughput in a short time to overload the controller and render its resources unavailable to legitimate users.

Another topic of vital importance regarding SDN is the security of Northbound Interface. This interface work as a bridge between the applications and the SDN controller, through programmers can extract device information, define new routing rules, collect statistical data, routing table, etc. The lack of proper "standardization" for this interface creates breaches to benefit malicious users to access the controller such as lack of proper authentication, malicious application usage, and misconfigurations. Therewithal, the use of applications from different third parties could be also a point to be attentive. Deployment of third-party applications containing bugs, malware, and not well designed could risk the security of the controller plane and affect the network infrastructure. Detection mechanisms are necessary to discover anomalies in advance. Indago [40] detects if an application is malign or benign based on security-sensitive APIs. In [41] author classifies an application as malicious using system

calls as features to the model. Although it still presents some limitations, SDN-GUARD [42] offers an alternative to combat rootkits

5.1.2 OpenFlow vulnerabilities

OpenFlow is the most used protocol to guarantee interoperability between the SDN controller and the infrastructure layer. Regardless of its large adoption, OpenFlow presents vulnerabilities that open a path to security breaches. For example, in [43] a new Topology attack called "Free-port" takes advantage of a vulnerability during the port-description message exchange. This happens when no port update message is sent to the controller when a link fails or the port is free. An attacker can take advantage of the inactivity of the victim to send a crafted Packet-in to mislead the controller about the victim's position.

A saturation attack is a type of DoS targeting the data plane through packet flooding. This attack explores vulnerabilities during the OpenFlow switch rule creation event to exploit the switch's buffer capacity to degrade or simply dismantle the communication with other switches and overload the controller bandwidth capacity. For every incoming packet, OpenFlow switches match its header fields with a rule stored at the flow table. In case, there is no available match, the Packet-In message is sent to the controller to create a new flow rule to be updated in flow table. The attacker sends a high volume of packets with spoofed unknown IP to force the switch to trigger Packet-IN messages to the controller until exhausting its bandwidth link capacity. Furthermore, legitimate users are blocked to communicate with the switch under attack.

5.1.3 Literature gap

During the creation of this Thesis, it was noticeable that the quantity distribution of articles related to certain attacks was more prominent than the others. For instance, there is very limited literature regarding side-channel attacks in comparison to Dos attacks and Topology attacks. Consequently, it is difficult to visualize some points in the state-of-art solutions to address the security of the SDN control plane. It is expected that the void must be fulfilled with future brand new ideas to add value to future review articles.

5.2 Future Work

After the conclusion, some relevant points were raised that would be interesting to be considered in future works targeting academia or the industry. These points were raised partially from this work and other existing surveys to highlight the necessity of continuous improvement in this field.

5.2.1 Key Research points

- **Least Privilege principle applied to access control:**
  The lack of proper access modeling techniques to grant proper permissions to SDN applications can result in illegitimate or irresponsible access to the infrastructure layer. To address this specific issue, proper network access control policies must be endorsed to guarantee to reduce the risks of unauthorized access.
  One of the strategies to secure that privileges are not granted unintentionally is the least privilege permission. This principle states that the minimum authorization must be conceded to an

user to complete her/his task.

- **Proper Application-control authentication mechanisms :**
  The authentication mechanism certifies an SDN application's possession of proper credentials according to a database to be granted. In the previous chapters, it was noticeable that the solutions to the authentication of the SDN Northbound interface were extensive and diverse including the utilization of protocol IEEE 801.X, tokens, and digital signature.

- **Proper TLS authentication mechanisms :**
  During the discussion of the topic of Northbound interface authentication, it was noted the security issues related to a lack of standardization. In the case of the Southbound interface, authentication is optionally provided by the arbitrary adoption of the TLS/SSL protocol. However, this option still carries vulnerabilities during its authentication mechanism that can be exploited by attackers. Some articles attempt to solve this weakness with cryptography solutions and proposals to verify the steps during the handshake process [83, 84].

- **Use of proactive rules:**
  Proactive rules are computed on demand the flow rules without the dependency of a new flow rule insertion event. Floodguard [56] explored the concept of proactive rules as an option to reactive rules caused by table-miss of the current controllers. However, this method fails to prevent flooding attacks from happening since the attacker can explore a range of IPs that are not part of the network.

- **Packet-in Authentication :**
  Packet-in authentication is presented as a solution to protect the controller against spoofed packets that can trigger a saturation attack. This kind of attack utilizes a spoofing method to create false packets to trick the controller and switches. Consequently, it may be the starting point for a switch's buffer saturation attack, topology poisoning attack, Dos, etc. PacketCheck [87] and Inspector [88] share the same principle of authentication Packet-in messages to not allow exploitation.

- **Scalability, Consistency, and Resilience :**
  The major constraint of the proper operation of an SDN network to the scaling capacity of the controller due to the number of connections, the amount of data transferred and the number of switches supported. As a result, lack of scalability eases saturation flood attacks targeting OpenFlow-enabled switches that also affect the controller bandwidth capacity [2, 16].

  One suitable solution to approach scalability issues would be the employment of distributed controllers in diverse geographical locations as clusters. It is necessary to keep a high level of synchronization to maintain the operation. One of the biggest drawbacks of this design is the gain in terms of scalability, but potential losses in terms of consistency. A mismatch of information between the controller can not be propagated, creating a scenario for security breaches. Among the distributed controller's benefits in comparison to centralized architectures, is the resilience to failures in case of one the controllers are down.

- **Forensics Mechanisms :**
  Kreutz et al. [2] cited the lack of a trustful mechanism to perform a forensic analysis after an attack event at an SDN network. Forensics procedures permit the IT staff to investigate the causes of a problem or an attack to return the normal operations. The benefits of implementing forensics are positive for troubleshooting and debugging the source of the problematic event. Forensic researches are not yet mature, more investigation and research in this field are needed. [12, 16]

- **Network Automation :**
  Network Automation poses an alternative to countermeasure the complex environment of SDN networks maintain the dynamism and reliability required for a proper operation. At the introduction chapter, the capacity to respond to configure network traffic on demand and the lack of bounds to vendors due to the facility of the programmatic interfaces is one of the innumerable benefits of this technology. SDN flexibility accepts network automation tools to automatize tasks that could be previously manually performed reducing drastically the errors caused by human action. For this reason, it is of extreme importance the development tools can facilitate solutions deployment with maximum reliability.[12, 16].

6 Conclusion

This Thesis investigated and reviewed the most frequent security issues and attacks at the SDN control plane. Firstly, these issues were categorized according to the SDN controller perspective and mapped with the respective solutions currently available in the literature. Based on this idea, the attacks and threats were selected and categorized according to the impact on the SDN controller in two aspects: direct and indirect. For the direct impact the following options were grouped in clusters such as: Spoofing attacks, DDos attacks, SDN Northbound interfaces threats, Topology attacks, and Other attacks(attacks that could not be categorized in a specific group). In case of indirect attacks, the following clusters were formed: Fingerprinting attacks, SDN Southbound interface attacks, Reconnaissance attacks, and side-channel attacks.

By the simple definition, a direct impact targets directly controller resources, availability, confidentiality, integrity. The same applies to the indirect impact attacks, differently from the latter affects another component at the first stage, the effects are sensed by the controller at the end.

Based on this approach, literature research was carried out to cross these categorized attacks into the current solutions. Moreover, these solutions were again categorized to find similarities and patterns to be analysed. As an example, the permission solutions related to access control could be categorized in three aspects: Role-based, Behavior-based, and Parameterized-based. After the observation of these aspects, parameterized roles provide more granularity to the permissions and isolation of resources when an application access the SDN controller.

During this Thesis production, it was observed that some attacks could be grouped in one or two clusters. Moreover, the solutions addressing one specific cluster could be suitable for another one. For example, a solution to secure the TLS handshake process grouped in the cluster Authentication of Southbound Interface attack could also be clustered in the category Man-in-the-Middle of Southbound Interface attacks.

Finally, this Thesis ends with a discussion of relevant points discovered during the in-depth research of the articles and presentation of key-research points to be a further object of study. It is worth mentioning that some of the research points were brainstormed during this Thesis, while others are still open issues to be explored in the future.

References

[1] "Software-defined networking: The new norm for networks," Jun 2017.

[2] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," vol. 103, no. 1, 2015, pp. 14–76.

[3] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo, "Software defined wireless networks: Unbridling sdns," in *2012 European Workshop on Software Defined Networking*. IEEE, 2012, pp. 1–6.

[4] "The what and why of categories." [Online]. Available: https://berkeley.pressbooks.pub/td04p/chapter/the-what-and-why-of-categories/

[5] W. Stallings, *Foundations of Modern Networking*. Boston, MA: Addison-Wesley Educational, Oct. 2015.

[6] E. Z. N. Feamster, J. Rexford, "The road to sdn: An intellectual history of programmable networks."

[7] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, and M. Shaheed, "SDN security review: Threat taxonomy, implications, and open challenges," *IEEE Access*, vol. 10, pp. 45 820–45 854, 2022.

[8] F. Ubaid, R. Amin, F. Ubaid, and M. Muwar, "Mitigating address spoofing attacks in hybrid sdn," *International Journal of Advanced Computer Science and Applications*, vol. 8, 01 2017.

[9] D. L. . J. A. V. V. M. I. Z. Shu, J. Wan 2, "Security in software-defined networking: Threats and countermeasures," 2016.

[10] M. A. Aladaileh, M. Anbar, I. H. Hasbullah, Y.-W. Chong, and Y. K. Sanjalawe, "Detection techniques of distributed denial of service attacks on software-defined networking controller-a review," vol. 8, pp. 143 985–143 995.

[11] S. Khan, A. Gani, A. W. Abdul Wahab, M. Guizani, and M. K. Khan, "Topology discovery in software defined networks: Threats, taxonomy, and state-of-the-art," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 303–324, 2017.

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

[13] J. Xie, D. Guo, Z. Hu, T. Qu, and P. Lv, "Control plane of software defined networks: A survey," *Computer Communications*, vol. 67, pp. 1–10, 2015.

[14] B. Rauf, H. Abbas, M. Usman, T. A. Zia, W. Iqbal, Y. Abbas, and H. Afzal, "Application threats to exploit northbound interface vulnerabilities in software defined networks," *ACM Comput. Surv.*, vol. 54, no. 6, jul 2021.

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

[16] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *Commun. Surveys Tuts.*, vol. 18, no. 1, p. 623–654, jan 2016.

[17] M. B. Jiménez, D. Fernández, J. E. Rivadeneira, L. Bellido, and A. Cárdenas, "A survey of the main security issues and solutions for the sdn architecture," *IEEE Access*, vol. 9, pp. 122 016–122 038, 2021.

[18] Y. Liu, B. Zhao, P. Zhao, P. Fan, and H. Liu, "A survey: Typical security issues of software-defined networking," *China Communications*, vol. 16, no. 7, pp. 13–31, 2019.

[19] "Defintion vulnerabilities." [Online]. Available: https://csrc.nist.gov/glossary/term/vulnerabilities

[20] "Defintion threat." [Online]. Available: https://csrc.nist.gov/glossary/term/threat

[21] "Defintion attack." [Online]. Available: https://csrc.nist.gov/glossary/term/attack

[22] J. Li, S. Qin, T. Tu, H. Zhang, and Y. Li, "Packet injection exploiting attack and mitigation in software-defined networks," *Applied Sciences*, vol. 12, no. 3, p. 1103, Jan. 2022.

[23] S. K. N. J. Surya Maddu, S. Tripathy, "Sdnguard: An extension in software defined network to defend dos attack," 2019.

[24] T. Alharbi, D. Durando, F. Pakzad, and M. Portmann, "Securing arp in software defined networks," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, 2016, pp. 523–526.

[25] Y. Gautam, B. P. Gautam, and K. Sato, "Experimental security analysis of sdn network by using packet sniffing and spoofing technique on pox and ryu controller," in *2020 International Conference on Networking and Network Applications (NaNA)*, 2020, pp. 394–399.

[26] L. Yang and H. Zhao, "Ddos attack identification and defense using sdn based on machine learning method," in *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, 2018, pp. 174–178.

[27] K. Hong, Y. Kim, H. Choi, and J. Park, "Sdn-assisted slow http ddos attack defense method," *IEEE Communications Letters*, vol. 22, no. 4, pp. 688–691, 2018.

[28] M. Han, T. N. Canh, S. C. Noh, J. Yi, and M. Park, "Daad: Dns amplification attack defender in sdn," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, 2019, pp. 372–374.

[29] R. Neres Carvalho, J. Luiz Bordim, and E. Adilio Pelinson Alchieri, "Entropy-based dos attack identification in sdn," in *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2019, pp. 627–634.

[30] J. Mao, W. Deng, and F. Shen, "Ddos flooding attack detection based on joint-entropy with multiple traffic features," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 237–243.

[31] A. Al-Alaj, R. Sandhu, and R. Krishnan, "A formal access control model for se-floodlight controller," *SDN-NFVSec '19: Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, pp. 1–6, 03 2019.

[32] A. Al-Alaj, R. Krishnan, and R. Sandhu, "Sdn-rbac: An access control model for sdn controller applications," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, 2019, pp. 1–8.

[33] B. Toshniwal, K. D. Joshi, P. Shrivastava, and K. Kataoka, "Beam: Behavior-based access control mechanism for sdn applications," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, 2019, pp. 1–2.

[34] A. Al-Alaj, R. Krishnan, and R. Sandhu, "Parasdn: An access control model for sdn applications based on parameterized roles and permissions," in *2020 IEEE 6th International Conference on Collaboration and Internet Computing (CIC)*, 2020, pp. 107–116.

[35] H. Cui, Z. Chen, L. Yu, K. Xie, and Z. Xia, "Authentication mechanism for network applications in sdn environments," in *2017 20th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, 2017, pp. 1–5.

[36] Y. Tseng, Z. Zhang, and F. Naït-Abdesselam, "Controllersepa: A security-enhancing sdn controller plug-in for openflow applications," in *2016 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, 2016, pp. 268–273.

[37] A. L. Aliyu, A. Aneiba, and M. Patwary, "Secure communication between network applications and controller in software defined network," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, 2019, pp. 1–8.

[38] Y. E. Oktian, S. Lee, H. Lee, and J. Lam, "Secure your northbound sdn api," in *2015 Seventh International Conference on Ubiquitous and Future Networks*, 2015, pp. 919–920.

[39] D. M. F. Mattos and O. C. M. B. Duarte, "AuthFlow: authentication and access control mechanism for software defined networking," *Annals of Telecommunications*, vol. 71, no. 11-12, pp. 607–615, Mar. 2016.

[40] C. Lee, C. Yoon, S. Shin, and S. K. Cha, "Indago: A new framework for detecting malicious sdn applications," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, 2018, pp. 220–230.

[41] D. Chasaki and C. Mansour, "Sdn security through system call learning," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2021, pp. 1–6.

[42] D. Tatang, F. Quinkert, J. Frank, C. Röpke, and T. Holz, "Sdn-guard: Protecting sdn controllers against sdn rootkits," in *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2017, pp. 297–302.

[43] S. S. Baidya and R. Hewett, "Detecting host location attacks in sdn-based networks," in *2020 29th Wireless and Optical Communications Conference (WOCC)*, 2020, pp. 1–6.

[44] R. Nagarathna and S. M. Shalinie, "Slamhha: A supervised learning approach to mitigate host location hijacking attack on sdn controllers," in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, 2017, pp. 1–7.

[45] L.-D. Chou, C.-C. Liu, M.-S. Lai, K.-C. Chiu, H.-H. Tu, S. Su, C.-L. Lai, C.-K. Yen, and W.-H. Tsai, "Behavior anomaly detection in sdn control plane: A case study of topology discovery attacks," in *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, 2019, pp. 357–362.

[46] D. Smyth, S. McSweeney, D. O'Shea, and V. Cionca, "Detecting link fabrication attacks in software-defined networks," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–8.

[47] T. Alharbi, M. Portmann, and F. Pakzad, "The (in)security of topology discovery in software defined networks," in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, 2015, pp. 502–505.

[48] A. Azzouni, O. Braham, T. M. Trang Nguyen, G. Pujolle, and R. Boutaba, "Fingerprinting openflow controllers: The first step to attack an sdn control plane," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–6.

[49] B. Ahmed, N. Ahmed, A. W. Malik, M. Jafri, and T. Hafeez, "Fingerprinting SDN policy parameters: An empirical study," *IEEE Access*, vol. 8, pp. 142 379–142 392, 2020.

[50] B. Toshniwal, K. D. Joshi, P. Shrivastava, and K. Kataoka, "Beam: Behavior-based access control mechanism for sdn applications," in *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, 2019, pp. 1–2.

[51] M. Masoud, Y. Jaradat, and A. Ahmad, "On tackling social engineering web phishing attacks utilizing software defined networks (sdn) approach," 12 2016.

[52] S. Midha and K. Triptahi, "Extended tls security and defensive algorithm in openflow sdn," in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2019, pp. 141–146.

[53] S. S. Mahdi and A. A. Abdullah, "Improved security of sdn based on hybrid quantum key distribution protocol," in *2022 International Conference on Computer Science and Software Engineering (CSASE)*, 2022, pp. 36–40.

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

[55] S. N, A. K.V, and B. B, "Detection and mitigation of MITM attack in software defined networks," in *Proceedings of the First International Conference on Combinatorial and Optimization, ICCAP 2021, December 7-8 2021, Chennai, India*. EAI, 2021.

[56] H. Wang, L. Xu, and G. Gu, "Floodguard: A dos attack prevention extension in software-defined networks," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2015, pp. 239–250.

[57] Y. Tian, V. Tran, and M. Kuerban, "Dos attack mitigation strategies on sdn controller," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0701–0707.

[58] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "Lineswitch: Tackling control plane saturation attacks in software-defined networking," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1206–1219, 2017.

[59] D. Ono, S. Izumi, T. Abe, and T. Suganuma, "A design of port scan detection method based on the characteristics of packet-in messages in openflow networks," in *2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2020, pp. 120–125.

[60] J. Patil, V. Tokekar, A. Rajan, and A. Rawat, "Port scanning based model to detect malicious tcp traffic and mitigate its impact in sdn," in *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, 2021, pp. 365–370.

[61] A. Alshamrani, "Reconnaissance attack in sdn based environments," in *2020 27th International Conference on Telecommunications (ICT)*, 2020, pp. 1–5.

[62] F. Standaert, "Introduction to side-channel attacks," 2010.

[63] A. J. A. J. M. S. E. K. J. Sonchack, A. Dubey, "Timing-based reconnaissance and defense in software-defined network," 2016.

[64] M. Numan, F. Hashim, and N. A. A. Latiff, "Detection and mitigation of arp storm attacks using software defined networks," in *2017 IEEE 13th Malaysia International Conference on Communications (MICC)*, 2017, pp. 181–186.

[65] G. Darwesh, A. Vorobeva, and V. Korzhuk, "An efficient mechanism to detect and mitigate an arp spoofing attack in software-defined networks," *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, vol. 21, pp. 401–409, 06 2021.

[66] X. Zhang, L. Cao, Z. Meng, and X. Yao, "A solution for arp attacks in software defined network," in *AIIPCC 2021; The Second International Conference on Artificial Intelligence, Information Processing and Cloud Computing*, 2021, pp. 1–9.

[67] I. Amezcua Valdovinos, J. Pérez, K.-K. R. Choo, and J. Botero, "Emerging ddos attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions," *Journal of Network and Computer Applications*, vol. 187, p. 103093, 05 2021.

[68] S. Kottler, "February 28th ddos incident report," Feb 2018. [Online]. Available: https://github.blog/2018-03-01-ddos-incident-report/

[69] A. Aljuhani, "Machine learning approaches for combating distributed denial of service attacks in modern networking environments," *IEEE Access*, vol. 9, pp. 42 236–42 264, 2021.

[70] K. Singh, P. Singh, and K. Kumar, "Application layer http-get flood ddos attacks: Research landscape and challenges," *Computers & Security*, vol. 65, pp. 344–372, 2017.

[71] P. Maity, S. Saxena, S. Srivastava, K. S. Sahoo, A. K. Pradhan, and N. Kumar, "An effective probabilistic technique for ddos detection in openflow controller," *IEEE Systems Journal*, vol. 16, no. 1, pp. 1345–1354, 2022.

[72] T. Ubale and A. K. Jain, "Srl: An tcp synflood ddos mitigation approach in software-defined networks," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, pp. 956–962.

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

[74] V. C. Hu, R. Kuhn, and D. Yaga, "Verification and test methods for access control policiesmodels," Tech. Rep., Jun. 2017.

[75] T.-H. Nguyen and M. Yoo, "Analysis of link discovery service attacks in sdn controller," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 259–261.

[76] S. Jero, W. Koch, R. Skowyra, H. Okhravi, C. Nita-Rotaru, and D. Bigelow, "Identifier binding attacks and defenses in Software-Defined networks," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 415–432.

[77] S. Shin and G. Gu, "Attacking software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM Press, 2013.

[78] S.-W. Han, H. Kwon, C. Hahn, D. Koo, and J. Hur, "A survey on mitm and its countermeasures in the tls handshake protocol," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2016, pp. 724–729.

[79] "Reconnaissance." [Online]. Available: https://dictionary.cambridge.org/us/dictionary/english/reconnaissance

[80] S. A. Shaikh, H. Chivers, P. Nobles, J. A. Clark, and H. Chen, "Network reconnaissance," *Network Security*, vol. 2008, no. 11, pp. 12–16, Nov. 2008.

[81] E. V. Ananin, A. V. Nikishova, and I. S. Kozhevnikova, "Port scanning detection based on anomalies," in *2017 Dynamics of Systems, Mechanisms and Machines (Dynamics)*, 2017, pp. 1–5.

[82] C. Banse and S. Rangarajan, "A secure northbound interface for sdn applications," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 834–839.

[83] H. L. Y. E. O. J.Lam, S. Lee, "Securing distributed sdn with ibc," 2015.

[84] P. S. T. A. A. Ranjbar, M. Komu, "An sdn-based approach to enhance the end-to-end security: Ssl/tls case study," 2016.

[85] B. T. B. Yigit, G. Gür and F. Alagöz, "Secured communication channels in software-defined networks," 2019.

[86] J. M. B. M. J. D'Orsaneo, M. Tummala, "Analysis of traffic signals on an sdn for detection and classification of a man-in-the-middle attack," 2018.

[87] S. Deng, X. Gao, Z. Lu, and X. Gao, "Packet injection attack and its defense in software-defined networks," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 695–705, 2018.

[88] A. S. Alshra'a and J. Seitz, "Using inspector device to stop packet injection attack in sdn," *IEEE Communications Letters*, vol. 23, no. 7, pp. 1174–1177, 2019.

[89] A. Toh, "Azure ddos protection-2021 q3 and q4 ddos attack trends." [Online]. Available: https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/

Acronyms

3GPP Third Generation Partnership Project. 2

AAA Authentication, authorization, and accounting. 14

ACL Access Control List. 1

API Application Programming Interface. 5–9, 11, 13, 27, 28

ARP Address Resolution Protocol. xi, 19, 20, 22, 23, 31–33

CA Certificate Authority. 50

CPU Central Processing Unit. 18, 31

DDoS Distributed Denial of Service. ix, 12–15, 19, 20, 23, 24, 35, 36, 51, 54, 57

DNS Domain Name System. xi, 19, 24, 27, 31, 36, 37

DoS Denial of Service. ix, 14, 15, 18, 19, 23, 35, 36, 57, 58

DWT Discrete Wavelet Transform. 51

HOIC High Orbit Ion Cannon. 24

HTS Host Tracking system. 29

HTTP Hypertext Transfer Protocol. 24, 25, 31, 71

HULK HTTP Unbearable Load King. 24

ICMP Internet Control Message Protocol. 25–27, 34, 35

IEEE Institute of Electrical and Electronics Engineers. 2

IETF Internet Engineering Task Force. 2

IP Internet Protocol. 19, 22–29, 31, 33, 34

ITU International Telecommunication Union. 2

LLDP Link Layer Discovery Protocol. xi, 15, 29–31

MAC Media Access Control. 19, 22, 23, 29, 31

MitM Man-in-the middle. xiii, 17, 18, 20, 32, 33, 43, 50, 51

NIST National Institute of Standards and Technology. 17

NOS Network Operating System. 7, 9

OFDP OpenFlow Discovery Protocol. 29

ONF Opening Networking Foundation. v, 1, 2, 6, 31

PKG Private Key Generator. 48

REST Representational state transfer. 8, 11

SDN Software Defined Networks. v, ix, xi, 1–3, 5–10, 12–15, 17–19, 22, 23, 26–29, 31, 35, 50, 51, 57, 59

SECAS Secure Communication Architecture for Software-Defined Networks. 50

SOM Self-Organizing Maps. 15

SSL Secure Sockets Layer. 10, 31, 33

TCP Transmission Control Protocol. 24, 25, 33

TLS Transport Layer Security. xi, 10, 14, 28, 31–33

UDP User Datagram Protocol. 23, 25–27

URL Uniform Resource Locator. 27

Acknowledgments

Thank you so much professor Chhagan Lal, member of the Spritz group for sharing your knowledge and competences.It was really a pleasure and I will bring the lessons learned to all my life.

I would also like to thank you professor Conti for this opportunity to develop this challenge. Grazie mille!