Journal of Network and Computer Applications 159 (2020) 102595

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Security in SDN: A comprehensive survey

Juan Camilo Correa Chica *, Jenny Cuatindioy Imbachi, Juan Felipe Botero Vega

Universidad de Antioquia and Instituto Tecnológico Metropolitano de Medellín, Universidad de Medellín, Universidad de Antioquia Calle, 67 # 53 – 108, Medellín, Colombia

ARTICLE INFO

Keywords:
Software defined networking
Network security
Attack detection
Vulnerabilities
Network monitoring
Traffic inspection
Openflow
Programmable networks
Network applications
Security threats
Threats mitigation
Virtualized network functions
Forensics

ABSTRACT

Software Defined Networking (SDN) is a revolutionary paradigm that is maturing along with other network technologies in the next-gen trend. The separation of control and data planes in SDN enables the emergence of novel network features like centralized flow management and network programmability that encourage the introduction of new and enhanced network functions in order to improve prominent network deployment aspects such as flexibility, scalability, network-wide visibility and cost-effectiveness. Although SDN exhibits a rapid evolution that is shaping this technology as a key enabler for future implementations in heterogeneous network scenarios, namely, datacenters, ISPs, corporate, academic and home; the technology is far from being considered secure and dependable to this day which inhibits its agile adoption. In recent years, the scientific community has been attracted to explore the field of SDN security to close the gap to SDN adoption. A twofold research context has been identified: on the one hand, leveraging SDN features to enhance security; while on the other hand one can find the pursue of a secure SDN system architecture. This article includes a description of security threats that menace SDN and a list of attacks that take advantage of vulnerabilities and misconfigurations in SDN constitutive elements. Accordingly, a discussion emphasizing the duality SDN-for-security and SDN-security is also presented. A comprehensive review of state-of-the art is accompanied by a categorization of the current research literature in a taxonomy that highlights the main characteristics and contributions of each proposal. Finally, the identified urgent needs and less explored topics are used to outline the opportunities and future challenges in the field of SDN security.

1. Introduction

The SDN paradigm decouples the control and data planes, therefore all network intelligence and control logic is migrated from the network devices to a logically centralized software-based entity known as the network controller. Network controller resides in the control plane where centralized control and network management functions instruct forwarding behavior to all the elements distributed in the infrastructure. In the data plane, network elements called switches are designed to match metadata in flow packets against the rules and forwarding instructions issued by the network controller. Such process runs before conceiving any forwarding decision. The centralized characteristic of SDN implies that the network controller is always aware of the network state and that all traffic flows are passed to the controller at least once in the network lifetime for the definition of forwarding behavior. Besides centralized flow management, SDN nurtures the conception of network programmability; hence, different network functions are embedded as software applications that can be either installed on top of the controller or deployed as independent data consumer functions.

SDN embodies the concept of network programmability since all network operations must be described as software programs, integrating algorithms, data structures and programming concepts that belong to the software development environment. Security, a sensitive aspect in communication and data networks, may benefit from SDN features including the network programmability itself; several security problems that often threat conventional networks can be sorted out in SDN in a timely and reliable manner enforcing network security software applications. The introduction of security applications to networking environments might contribute to: replace hardware-based security middleboxes; add extra security features by the integration of emergent software-based technologies to the SDN environment, such as machine learning and virtualization; redefine, improve or sanitize conventional

* Corresponding author.
E-mail addresses: camilo.correa@udea.edu.co , juancorrea@itm.edu.co (J.C. Correa Chica), jecuantindioy@udem.edu.co (J.C. Imbachi), juanf.botero@udea.edu.co (J.F. Botero Vega).

https://doi.org/10.1016/j.jnca.2020.102595
Received 13 December 2018; Received in revised form 29 November 2019; Accepted 9 March 2020
Available online 19 March 2020

networking mechanisms and concepts with a proclivity to expose vulnerabilities or weak points; increase scalability and flexibility of security solutions and ease the deployment of such solutions in massive network infrastructures. Despite all the benefits listed before, it is convenient to highlight that the SDN architecture entails unknown and additional security problems, risks and threats that emerge due to the introduction of new network interfaces and the alteration of network elements and their traditional communication scheme. Therefore besides developing new security strategies leveraging SDN there must be a commitment to embed security in the reference architecture of the SDN itself.

As stated above, SDN can be leveraged to enhance network security but, at the same time, it needs to be precisely secured. Both approaches have attracted the attention of researchers considering that the SDN paradigm will not consolidate as the reference network architecture for enterprises and ISPs until a strict security scheme that cooperates with fulfilling a standardized set of minimum security requirements is completely devised. Back in 2008, the first security proposals for SDN appeared (Hinrichs et al., 2008) and since then to the recent years many different proposals have been published; some of them provide new network security mechanisms and improvements that leverage SDN features while some others focus on building a secure framework for reliable SDN deployment, just to name a few: PermOF (Wen et al., 2013) SE-Floodlight (Porras et al., 2015), FortNOX (Porras et al., 2012), Rosemary (Shin et al., 2014), LegoSDN (Chandrasekaran and Benson, 2014), Avant-Guard (Shin et al., 2013a), CPRecovery (Fonseca et al., 2012), NICE (Canini et al., 2012), FlowChecker (Al-Shaer and Al-Haj, 2010), Veriflow (Khurshid et al., 2012), FlowGuard (Hu et al., 2014a), Frenetic (Foster et al., 2011), Verificare (Skowyra et al., 2013), FRESCO (Shin et al., 2013b), NICE:NIDS (Chung et al., 2013), SnortFlow (Xing et al., 2013) and CBAS (Toseef et al., 2014). Those security proposals have been presented and thoroughly analyzed and discussed in comprehensive survey articles published throughout the recent years (Dacier et al., 2017, Ahmad et al., 2015, Scott-Hayward et al., 2016, Scott-Hayward et al., 2013, Akhunzada et al., 2015, Rawat and Reddy, 2017, Alsmadi and Xu, 2015, Ali et al., 2015, Shu et al., 2016, Akhunzada et al., 2016, Coughlin, 2014, Shaghaghi et al., 1804). Therefore, we consider that no significant contribution will result from further analysis of such works. However, new proposals have been thickening the state-of-art reflecting the evolution of SDN technology and covering newly discovered security aspects and details closely related to such evolution. Therefore, in this work we will focus on recently published SDN security works that introduce new security approaches obtained by either leveraging SDN features or by adapting to SDN security strategies that were developed for conventional networks.

The main goal of this work is to provide readers with a comprehensive revision of state-of-art proposals for the development and evolution of SDN security. In more detail, the main contributions of this survey are listed below:

1. A taxonomy that classifies the reviewed articles in two main categories: The first one groups the proposals that leverage SDN features to improve network security. While the other one marks proposals that provide solutions to intrinsic security faults and risks in SDN. Moreover, subsequent categories label the articles according to seven subclasses defined according to unique contributions and features exhibited by the proposals.
2. We foster a discussion to highlight two situations: The complexity in the integration of diverse security systems, and the Inconsistencies that can be observed in the design and implementation of security strategies for SDN. From our perspective and understanding those two situations need attention and additional efforts from the SDN community in order to ensure that the emerging security solutions overcome the problems and flaws present in current SDN security strategies.
3. An introduction to the open and emerging challenges in SDN security to suggest and point out the directions for future research efforts and proposals.

This work is structured as follows: Section 2 introduces a set of basic notions and concepts to provide a brief description of the SDN architecture and the OpenFlow protocol; in this section we also highlight network attacks, threat vectors and attack surfaces in SDN. In Section 3, we thoroughly analyze the duality in SDN security: enforcing SDN to improve network security or enhancing SDN security? In Section 4, the taxonomy and the classification are presented, along with a brief analysis of newly published proposals. In Section 5 we briefly detail some potential flaws and inconsistencies that are still present in the design of security mechanisms for SDN. In Section 6, we identify open challenges and future work in SDN security. And finally, in Section 7 we conclude this article.

2. Background and context

SDN paradigm decouples control and data planes, it means that all network logic and control is isolated from data devices. Although SDN proposes novel and promising network architectures, it is still in an early stage for realistic and production-like implementations. One key aspect that inibits the adoption of SDN as the de-facto network architecture is the security, there is still too much to develop and test regarding this aspect. Nevertheless, in recent years SDN security has been gaining more interest from the academy and the industry and there is a significant grow in the research efforts and publications regarding this matter. Sections 3 and 4 in this work go deeper into the details of SDN security and the proposals published up to these days. But before reaching those sections our purpose is to give a brief context regarding SDN architecture and its singularities, and then make a review of generalized network security aspects. And in that sense detail the attacks and security issues that are identified in SDN, describing and classifying them in a layered fashion according to the layers and interfaces that shape the SDN architecture.

2.1. SDN architecture and OpenFlow

2.1.1. SDN architecture

The most representative characteristic regarding Software Defined Networking is the decoupling of network control and packet forwarding tasks; it basically refers to the migration of all network intelligence, originally residing in hardware infrastructure to a logically centralized software-based entity while all forwarding devices become simple packet forwarding elements. Control and data planes decoupling in SDN, implies logical centralization of the control and management of all network forwarding devices which in turn promotes network management as a network-wide activity. (Fundation, 2012, Nunes et al., 2014, Kreutz et al., 2015).

Networking also benefits from planes decoupling and software programmability since complex network functions can be implemented using simplified software routines and algorithms. Then, different behavior or network functionality can be instructed to commodity infrastructure hardware (Kim and Feamster, 2013, Sezer et al., 2013, Feamster et al., 2013).

Programmability of the SDN controller offers network operators and customers comprehensive programming interfaces that can abstract all infrastructure low-level details (Lara et al., 2014a) enabling the possibility to simplify the enforcement of network forwarding behavior and policies, through more expressive high-level policy languages instead of using proprietary protocols or vendor-specific command sets.

Three major parts compose SDN architecture (see Fig. 1): Application, Control and Data planes, each one of these components exposes its own functional sub-layers and interfaces. Application plane holds

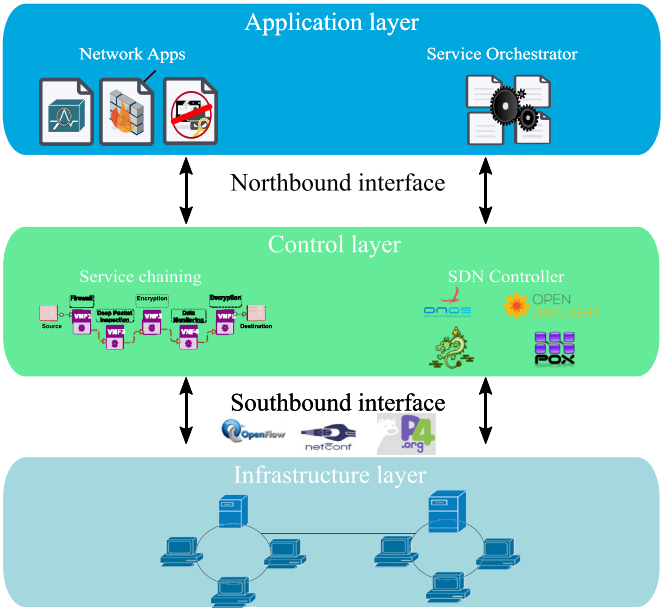

Fig. 1. SDN Architecture and Components: This figure depicts the components that reside at every single layer in the SDN architecture. For instance: custom and third-party applications in the application layer, SDN controllers in th control layer and infrastructure devices in the data plane.

the applications that define network behavior, policies and packet forwarding schemes. Control plane plays the role of Network Operating System where network controllers abstract low-level network details for network and management applications and also translate high-level policies into forwarding rules that are spread across the whole network infrastructure (Kreutz et al., 2015). Data plane refers to the set of forwarding devices composing network infrastructure, whose main functions are to enforce forwarding actions on flow packets according to corresponding instructions provided by the controller, and to report network status measures when requested by network applications. Northbound interfaces bind applications in Application plane to the Network Operating Systems (NOS) in the control plane, while southbound interfaces provision the control channel for data exchange between NOS and data plane devices (Li and Liao, 2013, Farhady et al., 2015, Jarraya et al., 2014).

2.1.2. OpenFlow

OpenFlow is the most widely deployed southbound interface or control layer protocol for SDN (Hu et al., 2014b, Braun and Menth, 2014, Pfaff et al., 2014). This protocol facilitates the communication between network applications that reside in the control and application layers, and all devices lying in the data forwarding layer (Vaughan-Nichols, 2011, Lara et al., 2014b, McKeown et al., 2008). The original OpenFlow draft specification has been constantly evolving and it now includes new features to face the emerging challenges imposed by the ever-growing SDN paradigm (Bianco et al., 2010, Benton et al., 2013).

OpenFlow provides a physical channel that is described in the specification as a dedicated TCP connection between controllers and switches that works as an interface for the exchange of control messages with their own format according to the OpenFlow protocol specification and with an optional encrytion mechanism that uses Transport Layer Security (TLS) (Pfaff et al., 2014, Lara et al., 2014b, McKeown et al., 2008, Benton et al., 2013, Fernandez, 2013). The nuts and bolts of the protocol specification describe 3 types of control messages that are used for the management of switches, and also for reporting of network events, and last to check the availability and state of control channel links (Stallings, 2013).

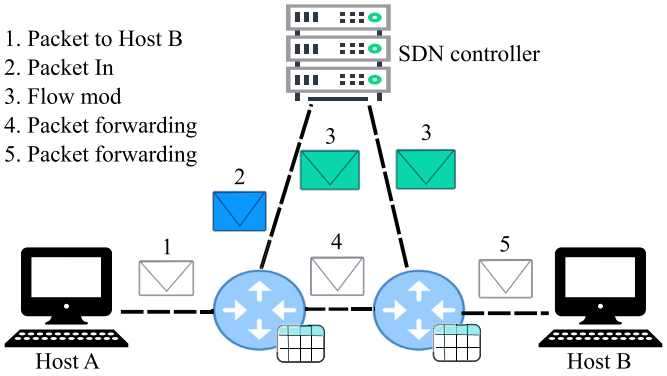

Fig. 2. Packet forwarding in OpenFlow: Steps 1 to 5 illustrate the process followed as per the OpenFlow specification to send a packet from Host A to Host B. In step 3 the switch asks the controller forwarding instructions and in step 4 the controller installs new forwarding rules in the switch.

To implement packet forwarding duties in SDN, using OpenFlow as southbound interface, switches have to support flow entries stored in flow tables. The flow entries are issued by the network applications and the controller, and are meant to instruct the packet forwarding behavior that is finally achieved by a process of matching packet fields and other features like priorities and packet counters against some constraints specified in the flow entries (Jarschel et al., 2011). A match between an incoming packet and a flow entry happens when all fields in the flow entry match all fields contained in the packet header. After a successful match, one out of six instructions included in the flow entry might be executed. Afterwards, certain action is executed over the packet according to the instruction (Stallings, 2013).

Right after the process of matching fields ends, a data packet might either be forwarded or dropped by the switch. While a control packet might either instruct the switch to install or remove a flow entry or might even issue a table-miss event if the packet did not match any of the flow entries currently stored in the flow tables (Stallings, 2013, Jarschel et al., 2011).

Steps for packet forwarding in OpenFlow based SDNs are briefly sketched in Fig. 2.

2.2. Threats and vulnerabilities in SDN

SDN as any new technology comes with its own pros and cons. For instance, and regarding security, SDN technology can be leveraged for relieving or totally mitigating some risks and vulnerabilities that are commonly exploited in conventional networks. Unfortunately SDN technology introduces new vulnerabilities and threat vectors that are inherent to its novel architecture. In fact, the separation of control and data planes, and the logical centralization of all network intelligence expose a single point of failure that can be exploited to compromise an entire SDN network. In the first part of this section, we provide a generalized overview of the most visible attack surfaces and threat vectors that have been identified in the planes and interfaces of the SDN architecture (Hogg, 2014, Kreutz et al., 2013; Tootoonchian etal., 2012). In the middle part, there is a detailed description of the issues, effects, and consequences derived from the exploitation of the attack surfaces and threat vectors present in SDN architecture. And in the final part, there is a list of most common attacks and malicious behaviors that target layers in SDN architecture.

2.2.1. Attack surfaces and threat vectors in SDN

Just like conventional networks; every network instance, protocol, device or layer participating in SDN can be subject to either intentional or innocent misuse that in some cases{ }is leveraged to either expose

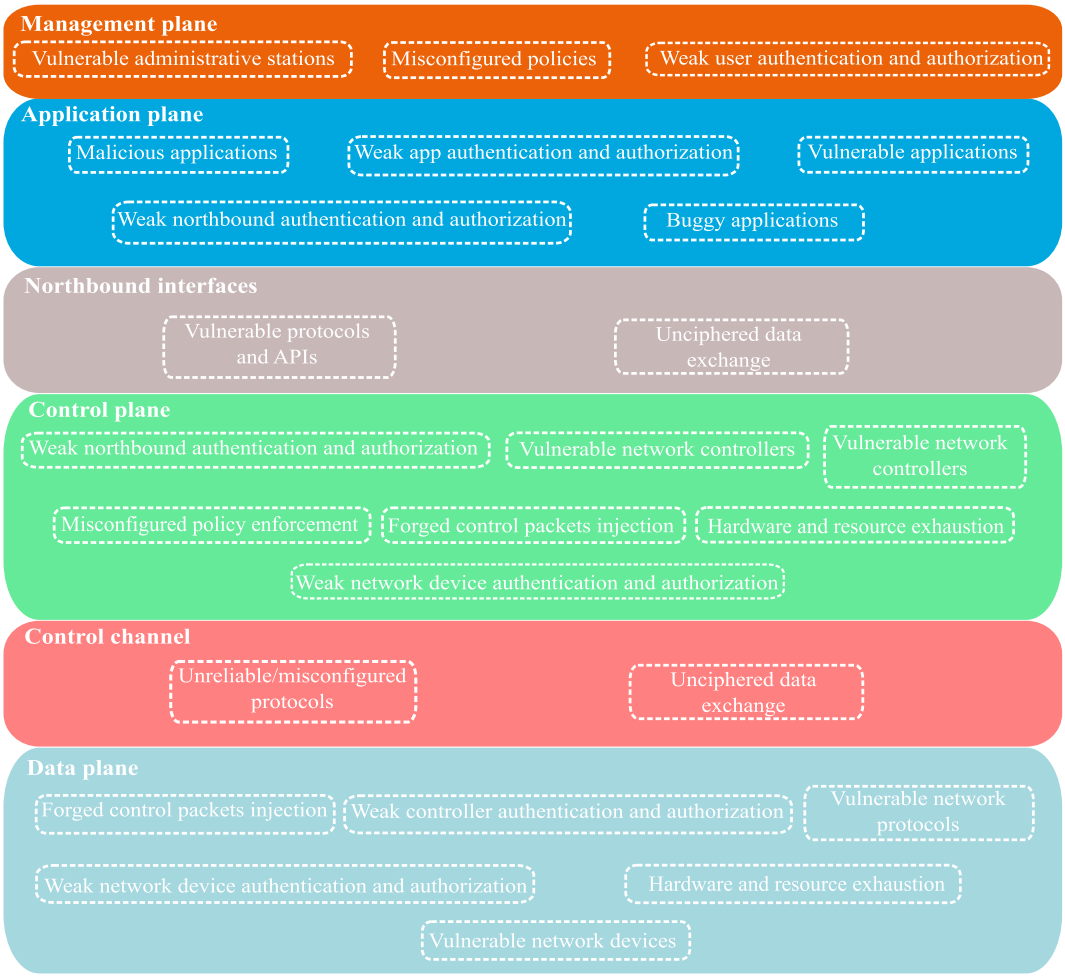

Fig. 3. Threats and vulnerabilities in SDN architecture: This figure shows the attacks and threats identified at all planes and interfaces in SDN architecture. Moreover, some attack surfaces and threat vectors with origin at certain plane/interface are often used to compromise targets in a different plane/interface.

system faults or to unveil implicit deviant behavior. This argument is sufficient to assert that every element or layer making part of the SDN architecture is a potential threat vector or attack surface, that is, any bad configuration or improper deployment of any SDN element can be concreted as an emerging source of vulnerabilities and potential security risks.

From this point onwards we are going to consider the control plane as a junction of both the Application layer and the Network Operating System layer, following this approach the vulnerabilities and attacks targeting network applications or northbound APIs will be considered control plane vulnerabilities or attacks without distinction. In Fig. 3, each SDN plane/interface groups a list with the most relevant threat surfaces that might be leveraged by malicious users to compromise a network under the SDN paradigm.

2.2.2. Security issues in SDN architecture

All attacks to the network security can be categorized according to the main objective of the attack, for instance eavesdropping on a control interface can be labeled as an attack whose main goal is to tamper with private and sensitive data exchanged between network appliances, it represents an unauthorized disclosure of information in a wider sense. The following list describes generalized issues and threats in network security by explaining the attack surfaces, deviant behaviours and security flaws that have been identified in the SDN context.

Unauthorized access (Sloan and Warner, 2013, Lindqvist and Jonsson, 1997): Attackers grant themselves unsupervised access to SDN elements exploiting either weak or inexistent access control mechanisms, launching brute-force attacks against administrative terminals and REST APIs that expose logging sessions, or exploiting vulnerabilities in network components and then installing rogue devices or binding remote connections to the network (Scott-Hayward et al., 2016, Hizver, 2015).

Unauthorized disclosure of information (Lindqvist and Jonsson, 1997, Jouini etal., 2014a): Several attack surfaces can be used to subtract sensitive network information, for instance, an attacker can infer about network forwarding behavior by broadcasting probe packets to target network elements. An attacker that successfully compromises a vulnerable network application can get access to network policy databases and other internal network data stores. Insecure channels can be seized for packet sniffing and eavesdropping. Finally, device impersonation attaks enable an attacker to receive information that is originally intended to a compromised network element (Pfaff et al., 2014, Benton et al., 2013, Hizver, 2015, Yoon et al., 2017a).

Unauthorized modification of network information (Lindqvist and Jonsson, 1997, Loch etal., 1992): Attackers can leverage vulnerable protocols and APIs and the lack of verification and authentication mechanisms to override existing flow rules through malicious applications issuing conflicting flow rules. Besides, an unauthorized access to internal storage offers an attacker the possibility to introduce contradictory network policies or modify the existent ones. On the other hand, changes in network topology can be induced leveraging protocol misconfigurations along with device impersonation and falsified packets

injection attacks (Scott-Hayward et al., 2016, Benton et al., 2013, Hogg, 2014, Kreutz et al., 2013, Hizver, 2015, Yoon et al., 2017a, Nguyen and Yoo, 2017, Röpke, 2016, Dover, 2014, Hong et al., 2015).

Destruction of network information (Lindqvist and Jonsson, 1997, Jouini etal., 2014a): The most prominent scenarios in this case refer to inflicted flow rule flushing in switches, malicious applications discarding control packets intended for either another application or a service chain, and network policy removal as a consequence of granted unauthorized access to either administrative stations or internal network databases (Scott-Hayward et al., 2016, Benton et al., 2013, Hizver, 2015, Yoon et al., 2017a).

Service disruption (Lindqvist and Jonsson, 1997, Jouini etal., 2014b): We identified three main sources of service disruption, in the first place the flooding attacks: control packet flooding and flow rule tables flooding. Secondly, the malformed packet injection attack that often causes connection hang-up or target device shut-down. Finally, topology poisoning attacks where controller network view is tricked, resulting in network disconnection of target devices (Yoon et al., 2017a, Dover, 2014, Dover, 2013).

Misconfigurations (Kendall, 1999, Simmons etal., 2014): Faulty configurations in protocols, interfaces, APIs and systems are translated into new vulnerabilities and security breaches; conflicting network policies and flow rules; improper verification and authentication mechanisms; and circumvention to already deployed security measures (Benton et al., 2013, Nguyen and Yoo, 2017, Röpke, 2016, Dover, 2014, Röpke and Holz, 2015).

Poorly configured authentication, trust and verification mechanisms (Simmons etal., 2014): Plain text channels and improper or weak authentication mechanisms pave the way for the vast majority of attacks and deviant behavior: eavesdropping, packet injection, packet crafting, traffic interception, network information poisoning and identity hijacking (Benton et al., 2013).

2.2.3. Attacks to the SDN architecture

All layers and interfaces are sensitive to certain specific attacks that might either compromise network components residing in the layer itself or target elements in another layer. The following paragraphs list all the layers of the SDN architecture and provide the set of most common attacks identified in each layer. Each attack is portrayed by its origin (cause), for instance: attack surface or threat vector.

i) Application layer

Application termination by abusing fixed privileges and authority: Third-party and control applications with unrestricted authority in the network system can be compromised to encompass the execution of system commands that are mostly used to disconnect/shutdown certain sensitive network APIs or applications (Hogg, 2014, Yoon et al., 2017a, Röpke, 2016, Röpke and Holz, 2015).

Service neutralization: Malicious applications successfully installed on top of the controller can be used to manipulate control packet handlers, then execute a service disruption by 1) discarding the control packets to prevent them from reaching the applications they are intended for; 2) subverting the order in which application handlers access control packets; 3) interfering in service chains to disrupt control packet forwarding; 4) inspecting control packets to sniff sensitive network information and execute specific deviant actions accordingly (Hogg, 2014, Yoon et al., 2017a, Röpke, 2016, Röpke and Holz, 2015).

Attacks to vulnerable northbound APIs: Misconfigurations and vulnerabilities in northbound APIs can be leveraged to either terminate a victim application by issuing a system command or to expose information exchanged between the controller and a target application (Hogg, 2014, Yoon et al., 2017a, Röpke, 2016, Röpke and Holz, 2015).

ii) Control layer

Dynamic flow rule tunneling: Attackers might circumvent firewall-like (block, drop) flow rules if they run malicious applications that are able to instruct overlapping and conflicting flow rules, taking advantage of the fact that controllers cannot distinguish implicit conflicts between newly issued rules and the existing ones that are bind to the different sets of network and control policies (Röpke, 2016).

Controller poisoning: Vulnerable network protocols and buggy/malicious applications can be used to poison controller information and the topology view which in turn propitiates the execution of attacks on the data plane. For instance, in the LLDP (Link Layer Discovery Protocol) packet injection attack, an adversary sends crafted LLDP packets to the controller to poison its network topology view since these tampered packets induce the controller to add fake network link entries to its own topology records. Another example is the Host Location Hijacking attack, in which a vulnerability in the Host Profile Service can be leveraged using crafted LLDP packets, as a result the controller host profile reservoir is poisoned and packets intended for a particular target switch are hereafter redirected to the attacker host (Yoon et al., 2017a, Nguyen and Yoo, 2017, Hong et al., 2015).

Network Operating System misuse: Compromised applications and rogue data-plane devices can exploit controller vulnerabilities and misconfigurations to achieve different objectives, for example, the execution of a system command that forces controller terminatio; leakage of sensitive information that resides in any instance of internal network data storage; redirection of information intended for a legitimate device; network policies databases hijacking; installation of rootkits or remote access connections to maintain an unauthorized access channel to the controller; and finally the introduction of invalid input data that might leave the controller in an unpredictable state (Röpke, 2016).

Packet-in flooding: By means of single or distributed compromised hosts or/and switches an adversary broadcasts massive malformed network packets that network infrastructure recipients translate into packet-in messages to the controller due to the occurrence of a considerable percentage of switch table misses. The controller might waste all of its computational resources by responding to the vast amount of packet-in packets coming in through its south-bound interface (Benton et al., 2013).

Controller's switch table flooding: The absence of a mechanism for authentication and verification of the identity of the sender of incoming packets in OpenFlow control packet reception at the controller makes room for different attack surfaces. Switch table flooding is the result of an uncontrolled behavior that is a response to the continuous reception of unexpected forged "features-reply" messages. This situation permits an attacker to store entries with information about fake switches in the target controller's switch table. Continuous execution of this attack results in degradation of controller performance due to the constant flooding of its corresponding switch table (Benton et al., 2013).

Forced switch disconnection: A controller can be forced to disconnect a legitimate switch if an attacker is successful launching any of the following attacks: 1) legitimate switch identity hijacking: In first place, an attacker impersonates a legit switch using its unique DPID (Datapath Identifier) and then tries to establish a connection to the controller. A successful fake switch connection results in the controller dropping the connection to the legit switch, genuine owner of the DPID. After disconnection, the target switch usually recovers the connection and the fake switch is disconnected by the controller, so the attack is fully launched if the connection/disconnection cycle is fully automated by the attacker, which leaves the target switch trapped in such hindering situation; 2) Spanning Tree poisoning: an attacker can fabric fake redundant links to poison the Spanning Tree Protocol using crafted LLDP packets, this maneuver tricks the controller and forces it to disconnect the pretended redundant links that the attacker has cleverly targeted. These links usually connect target hosts and switches; 3) deploying in the network operating system a malware that is able to corrupt the controller switch table information (Benton et al., 2013, Hong et al., 2015, Dover, 2013).

iii) Control channel

Eavesdropping: An adversary can leverage unencrypted control channels to perform packet sniffing attacks, then the attacker is able to listen to all control, topology and management information of the network that is exchanged in the control channel (Benton et al., 2013, Yoon et al., 2017a).

Man in the middle: Unciphered control channels along with ARP poisoning attacks can be leveraged to insert an intruder host in the middle of the control channel to infiltrate communications between the controller and any target data plane device (Benton et al., 2013, Yoon et al., 2017a).

iv) Infrastructure layer

Denial of service leveraging ARP poisoning attack: An attacker can achieve target switch isolation by the impersonation of the controller. Using the ARP poisoning attack, the attacker hijacks the controller's identity and forces a target switch to drop the connection to the genuine controller and connect to the fake controller instead. This results in achieving switch disconnection to the network (Yoon et al., 2017a, Hong et al., 2015).

Flow-rule modification/flushing: Attackers can tamper with the information installed in the switches' flow-tables, either overwriting or flushing existing flow-rules. Attackers can launch this attack from either a compromised application or a compromised network controller. (Yoon et al., 2017a).

Flow-rule flooding: By means of side-channel attack techniques an attacker can make inferences about two particular situations: 1) if a switch table is close to be completely filled (or full in fact) and 2) the kind of packets that generate a table miss, forcing the switch to raise a request to the controller for installation of a new flow rule. Given the situation in which the attacker has inferred about such information, then he/she is able to launch a flow table flooding attack forcing the switch to constantly ask for new rules and then fill its flow table, such behavior can have negative effects on switch performance and stability (Yoon et al., 2017a).

Malformed control packet injection: A target switch can be driven to an undesirable state if it gets exposed to a fuzzing-attack situation, receiving crafted control packets containing malformed or misused headers that are cleverly forged to expose existing vulnerabilities or misguided behavior experimented under invalid input circumstances (Yoon et al., 2017a).

Side-channel attacks: Also known as reconnaissance attacks usually leverage the resulting reactions of a particular target device against specific network situations to infer about certain implicit information that an attacker can later use to launch another kind of attacks. For instance, an attacker might record the RTT (Round-Trip Time) experimented by a particular packet when it has been sent to a particular switch, such information could be later leveraged to launch a flow table flooding attack on that switch (Yoon et al., 2017a).

Table 1 makes a summary of the attacks that can be launched against SDNs emphasizing on the sources and targets of the attack in order to establish a cause-effect relation between a punctual source of an attack and all the components of the SDN architecture that can be targeted. Such a relation might be helpful in determining levels of severity of damage that can be attributed to certain attacks, having as a measure for the severity the amount of SDN layers or interfaces that result compromised by launching the attack.

3. SDN for security and SDN security

SDN features like network-wide visibility, centralized network intelligence and network programmability reshaped the way packet forwarding and basic network control duties are performed in programmable networks. However, as detailed in the previous section, these features and the SDN architecture itself introduce new security risks and attack surfaces that are not present in conventional network deployments. According to the latter statement and taking on count the benefits to network control and packet forwarding that can be exploited from the SDN features, it can be seen that SDN security has a two-fold connotation: in the first place the exploitation of SDN features and mechanisms to protect, react and provide mitigation schemes against well-known security risks, either through the introduction and application of new security proposals or rendering the functionality of existing security systems and appliances, all this is described in the first part of this section. And in second place, the design of a secure SDN architecture aims at providing a proactive behavior against the new attack surfaces and security breaches introduced by SDN architecture itself (Yan et al., 2016). The latter is covered in the last part of this section.

3.1. Improving security through SDN

Three relevant SDN features that can be used to implement a variety of security solutions in the SDN environment are suggested in (Shin et al., 2016). In the following paragraphs, we briefly describe such features an introduce how they might be leveraged to improve security in SDN.

The first feature corresponds to the Dynamic Flow Control which is claimed to benefit the security in two different ways: 1) by enforcing the functionality of security middleboxes as a composition of different sets of flow control rules instructed throughout the network infrastructure; and 2) in the form of network applications either installed on top of the controller or bound to the controller through a northbound interface (see Fig. 4), so there is no need for additional hardware appliances that can be effectively replaced by embedding security rules in commodity network devices primarily intended for packet forwarding. Therefore, SDN dynamic flow control can be leveraged for the deployment of, to name just a few, perimeter and internal firewalls, comprehensive and ubiquitous access control lists and basic traffic redirection schemes. The latter mechanism introduces the second way in which dynamic flow control helps along with a traffic segmentation scheme that is used to distinguish between normal and suspicious traffic. Flow rules can be instructed dynamically to force network devices to redirect certain traffic types to more specialized security systems or to the controller itself for traffic analysis by a specialized security application.

The second one is a composition of Network-wide Visibility with Centralized Flow Control, both components explain the generalized holistic orientation of the SDN framework (Dacier et al., 2017), a property that reinforces the distinction between SDN and conventional network architectures. Network-wide visibility means that network control can be aware of the state of any network element deployed anywhere at any time. While centralized flow control refers to how flow forwarding decisions can be made from a single and logically centralized network instance; this behavior relieves every device in the network from making any computation or enforcing any forwarding algorithms when a data packet arrives. This composition of features can benefit several security tasks, in particular in all security aspects that involve the harvesting of intelligence, as the controller can request flow samples and flow statistics using specific messages implemented in the control layer protocol; collected information can be fed to specialized security applications, for instance, an IDS, IPS or DPI, or can even be directly used by the controller itself to instruct any kind of reaction according to the network situation inferred from gathered data.

By monitoring network traffic (Ahmad et al., 2015), the controller can keep updated records about traffic behavior, namely, sudden changes in both packet and byte level statistics at any network interface; unexpected traffic flows coming from unsupervised/supervised network elements; and suspicious or malformed traffic flows traversing the infrastructure. Attack detection and prevention, collected network features and network visibility enable the identification of the source of

Table 1
Attacks to SDN architecture (A: Application, NB: Northbound, C: Control, SB: Southbound, D: Data).

Attack	Sources	Targets				
		A	NB	C	SB	D
Abuse of privileges and authority (Röpke, 2016)	Vulnerable third-party applications	X	X	X		
Service disruption (Hogg, 2014)	Malware	X		X		
Application shutdown (Hogg, 2014)	Vulnerable northbound APIs	X	X	X		
Dynamic flow rule tunneling (Röpke, 2016)	Malware and vulnerable switches	X		X		X
Poisoned network view (Nguyen and Yoo, 2017)	Malware, vulnerable network services and protocols	X	X	X	X	X
NOS misuse (Röpke, 2016)	Vulnerable controller	X		X	X	X
Packet in flooding (Benton et al., 2013)	Faulty controller, compromised switches			X	X	X
Switch table flooding (Benton et al., 2013)	Faulty controller, compromised switches			X	X	X
Eavesdropping (Benton et al., 2013)	Unciphered control channel			X	X	X
Man in the middle (Yoon et al., 2017a)	Unciphered control channel, compromised southbound interface, vulnerable data links			X	X	X
Forced switch disconnection (Hong et al., 2015)	Compromised switches, vulnerable protocols					X
Flow table flooding (Yoon et al., 2017a)	Vulnerable switches			X	X	X
Switch shutdown (Hong et al., 2015)	Packet injection, fuzzing techniques					X
Switch exploitation (Hong et al., 2015)	Vulnerable switches					X

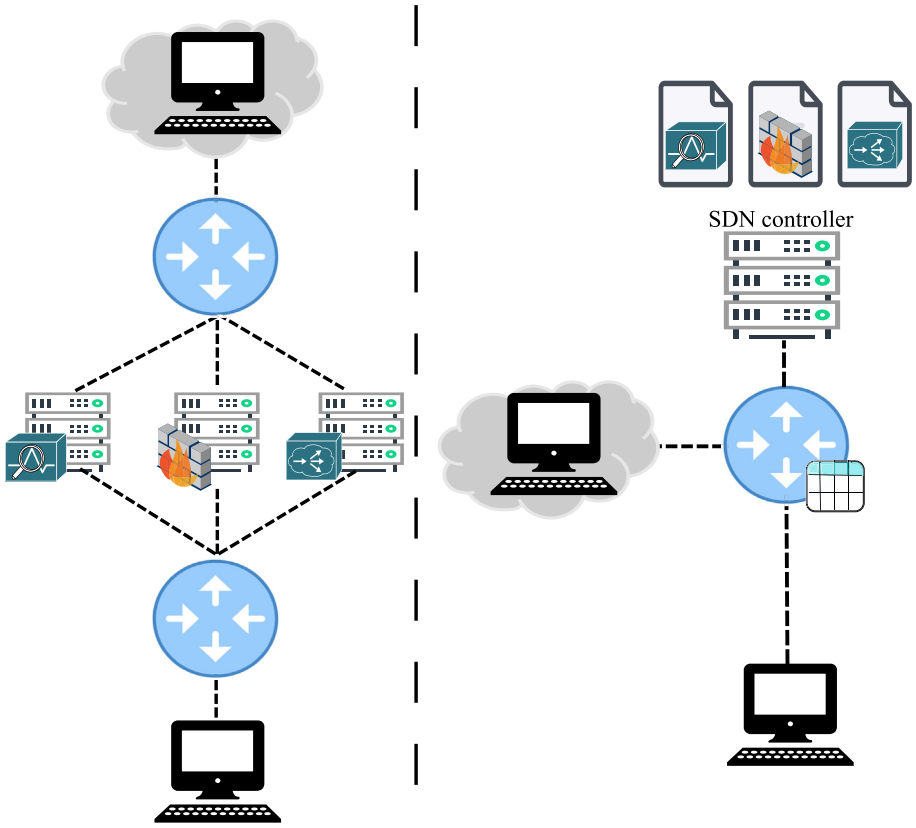

Fig. 4. Middlebox deployment in conventional networks (a) and in SDN (b). This figure depicts how hardware appliances connected to the network for different purposes can be replaced with software applications in SDN.

attacks, then the control plane can decide proper actions to mitigate the anomalous situation; for instance, flow statistics and traffic patterns can alert of a possible DoS (Denial of Service) attack, then the controller or certain security application has to invoke the measures defined to stop it, either installing new flow rules or completely denying any traffic from the attack source.

And finally, the flow content inspection, IDS, IPS and DPI can be found deployed in conventional networks in two flavors, as individual hardware appliances installed at fixed and neuralgic network locations, and even as software-based security applications running on multi-purpose servers. SDN enables the possibility to deploy these security systems in different ways, as controller applications, as security services residing in the application layer, as security services running in cloud environments and even as hardware appliances attached to the infrastructure. SDN can feed these systems redirecting traffic directly towards them or through features, information and packet samples extracted from the network state.

The third feature is network programmability (Haleplidis et al., 2015, Van der Merwe and Kalmanek, 2007, Kaur et al., 2014). In conventional networks, users/administrators are not allowed to modify or

alter network devices or applications at will. Instead, those devices come equipped with propietary command/instruction sets that only permit the execution of certain re-configurations and re-programming that are, in general, limited to a certain degree of customer intermediation fixed by the device or application producer. In SDN, due to network programmability the infrastructure can be made up of a set of commodity devices in which the customer can embed different functionalities, that are expressed through specific sets of flow rules. For instance, basic traffic filtering schemes and access control lists can be installed in network switches enforcing flow rules that express drop or deny actions against specific packet/flow features.

In SDN, network programmability enhances network flexibility, for instance, leveraging the southbound interfaces a network operator can parameterize the data plane according to the aspects that describe the context in which the network is operating (Sama et al., 2014). Network programmability also benefits the deployment of the functionality of a variety of security services and security systems as network applications either installed as controller applications or in a separate security applications layer. Replacing narrow and hard-to-update security appliances with flexible security software routines that can be designed to process a wide diversity of inputs, and to offer quick and adequate reactions to different attack situations caused by illegitimate network actors. Additionally, features and advantages provided by the most recent advances in software technology, artificial intelligence and machine learning can be leveraged to implement more robust security solutions capable of, for instance, predicting attacks by correlating different factors observed in the network behavior Also, these features might be able to execute counterattacks against network adversaries, for instance, a security network application that detected a malicious device/host can be equipped with algorithms and scripts that might release an attack to the malicious host with th goal of exhausting the attacker machine resources to stop the attack at least while the network gets shielded against it.

Network programmability introduced the capacity to reconfigure infrastructure devices or to deploy into them functionalities different to packet forwarding, for example basic security schemes. This refers to the notion of a Simplified Data Plane in SDN, meaning that the infrastructure is populated with commodity devices that can be modified at any time to cooperate with different network functions when required. This is the last SDN feature whose benefits to security we are going to expose in this work. In SDN the control and data planes are decoupled, so all device intelligence is extracted, thus a forwarding device has to rely on decisions instructed by a centralized entity, the network controller. Regarding security, this behavior can be leveraged to implement security functions and services according to certain security constraints, security policies or even certain security invariants introduced by the design of the security system itself.

A commodity switch, for instance, can be instructed to deny or drop certain data packets belonging to a specific traffic flow, in this manner basic access control lists and packet filtering schemes can be deployed taking advantage of the installed network resources and thus avoiding additional hardware deployment. Flow rules can also be used to instruct the infrastructure in the redirection of suspicious traffic flows to more specialized security systems, namely, security applications (either in the controller or in independent service layers), security appliances and security services residing in cloud environments. Other opinions suggest embedding virtualized security functions and virtualized security service chains in the infrastructure itself, this way the switches can be enabled to perform more fine-grained security functions, like traffic filtering and monitoring, packet inspection, threat detection and prevention, attacker isolation, and offloading the control plane with part of security processing (Matias et al., 2015, Battula, 2014, Deng et al., 2015).

It is clear that SDN characteristics can be exploited to benefit network security in different network contexts whether rendering and adapting well known security schemes or even introducing novel approaches to tackle existing security risks. However, as depicted in section 2, SDN architecture introduces new attack surfaces and security breaches, for instance, in traditional networks customers or adversaries do not have access to a communication channel between control and data planes since both are embedded within infrastructure devices, conversely, in SDN the control channel is a new layer in the network architecture, exposing new attack vectors and security risks not seen in conventional networks architecture. Having the latter in mind, it is important to point out that new security developments are needed to guarantee a secure SDN environment, what has been called system level SDN security that is going to be approached hereinafter.

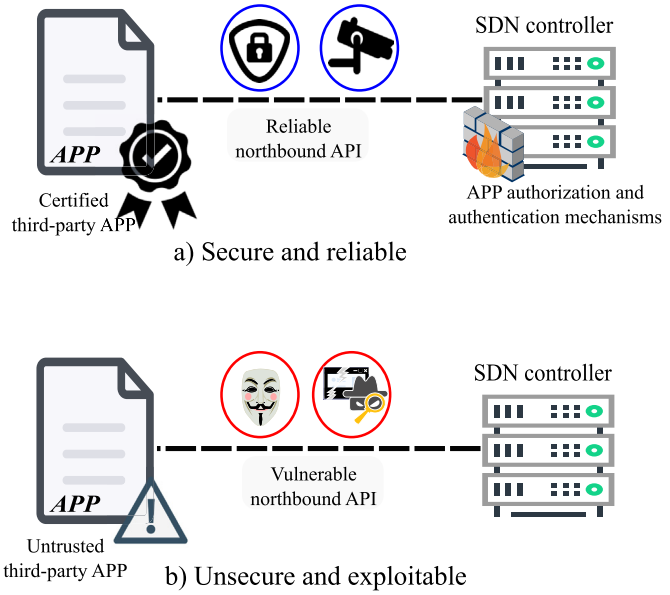

Fig. 5. Achieving a high level of protection and security in the boundaries of the control and application planes requires the development of different mechanisms for identifying and relieving the risks introduced by unreliable network applications and northbound interfaces. For instance: authorization and authentication schemes, assessment and security audits, and controlled access to management services and applications.

3.2. *Improving SDN security*

As previously mentioned in section 2, SDN architecture introduces new vulnerabilities and security risks due to inherent alterations to both the network components and the relationships between them (Scott-Hayward et al., 2016). Besides, the introduction of new interfaces and protocols also cooperates with the emergence of new attack surfaces and exploitable targets. Given these facts, an important conclusion comes up, despite the efforts to enhance network security leveraging SDN novelties and innovations, a network architecture might keep insecure if the SDN framework itself is not sanitized correctly against security threats introduced by SDN features.

Channels and interfaces used for information exchange in SDN may expose neuralgic network state communications when they are not properly secured, communications in plain text might be intervened and collected data used to compromise different network entities. Therefore, all SDN communication channels and data interfaces must be encrypted, this way an attacker is perhaps able to infiltrate ciphered data exchanges but is definitely unable to extract the real information that is being forwarded.

Along with encryption schemes, SDN security should be fortified using authentication and trust mechanisms especially at the control plane. The controller must be able to recognize and then authorize trusted devices (other controllers, switches and additional appliances) and network applications through exchange of either signed certificates, symmetric and asymmetric keys or message authentication codes

(Schehlmann et al., 2014). Ensuring that only trusted devices and applications have granted access to network resources guarantees that rogue devices and malicious applications remain isolated from the network (see Fig. 5). The selection of the protection mechanisms for the control channel or some other interface must be done carefully, having in mind the services and protocols that will use the channel. It has been proved that a wrong selection of the encryption mechanism results in exposing the channel to vulnerabilities (Liyanage et al., 2014).

A major fault present in SDN refers to the impossibility of the controller to detect network applications that might be issuing flow rules that conflict with the rules installed before by another application (or applications) or even flow rules that contradict pre-established security policies. As measures to mitigate such situations the SDN control plane should include a mediator scheme for conflicting rules (Porras et al., 2012, Schehlmann et al., 2014) and policy checking schemes as well (Dacier et al., 2016). The conflicting rules mediator might be used to identify rules that overwrite the policy enforcement behavior indicated by a prior set of rules, while the policy checker might guarantee that new flow rules ready to be issued do not conflict with pre-stablished security policies, thus violating the security scheme and exposing the security breaches supposed to be covered by the violated security policies. Implementing these solutions should help to mitigate attacks launched via malicious applications or deviant third-party applications, for instance, ill-intentioned flow rules issued by anomalous nested applications (script snippets) masqueraded in apparently harmless applications. Security kernels (Porras et al., 2015, Shin et al., 2014, Scott-Hayward, 2015), can help the controller in the execution of sanity-check assessments to the network applications, the security kernel might be used to detect faulty-applications or even detect risks that could emerge when a set of applications, that passed the sanity-checks individually, is used to compose service chains.

Different attacks targeting network state, for instance, network topology poisoning, denial of service, flow-rule insertion and rogue device infiltration might be detected and mitigated through robust network state monitoring mechanisms (Dacier et al., 2016) which offer the possibility to keep an updated record of current network state and behavior. Therefore, the network information collected at a certain time frame can be matched against a network configuration file or listing (containing flow-tables, topology information, list of authorized network devices, packet forwarding statistics, data and control interfaces state, etc.) to detect any inconsistencies in the network state, for instance, unexpected links between network devices, non-authorized hosts attached to the network, unusual traffic volumes traversing both data and control interfaces, flow-rules violating security policies, etc. Besides, network state monitoring could be leveraged to classify the different network states in either warning states, prone to failure or security risks, or stable states, this way the network can be provided with an adaption mechanism that upon historical network state information is able to predict further network states and avoid falling in states prone to failure, so the network is kept operating in failure-free and secure states.

Adoption of cloud environments and network functions virtualization (Bernardo and Chua, 2015, Banikazemi et al., 2013, Jain and Paul, 2013, Ordonez-Lucena et al., 2017, Omnes et al., 2015) may contribute to SDN security improvement by allowing the integration of a variety of security solutions offered by multiple vendors and security developers at expenses of introducing additional components and layers to the SDN architecture (see Fig. 6, for a detailed graphical description of the distribution of virtualized security entities throughout the SDN architecture). Virtualized security contributes to the scalability of SDN security as ubiquitous security sevices deployed in the cloud can be invoked and released anywhere in the topology, and both the security functions and applications can be shared and migrated between different security clouds. Additionally, network elements can be relieved of the extra processing that is needed to execute some complex and resource-consuming security functions.

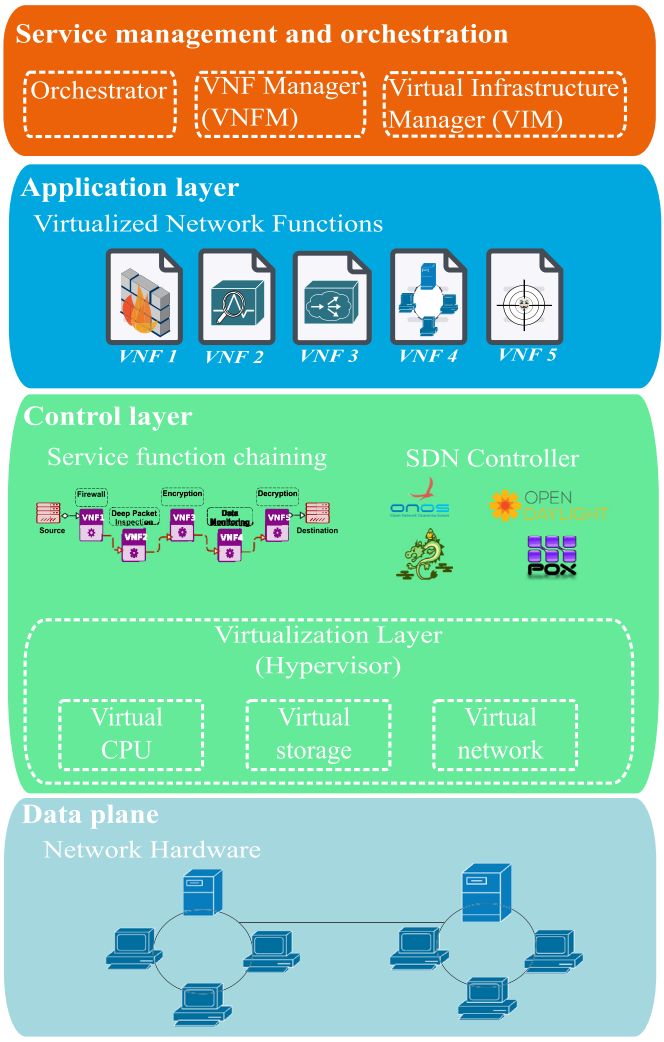

Fig. 6. Integration of NFV to SDN Architecture: Management and orchestration of Virtualized instances and interfaces are built on top of the application layer as an additional layer. Virtualized network functions reside, in general, in the application layers albeit some might reside in the data plane devices. Deployment, resources allocation, and control of virtualized components are control layer tasks. Missing in the figure are the corresponding links and interfaces that bind all components in the virtualized environment.

In the cloud-based security environments, the virtualized security functions might be created (instantiated) and destroyed on demand, this way a more robust security scheme can be built using a composition of basic and intermediate monitoring and detection security applications that can identify incipient attacks. Upon anomaly detection, this scheme might be able to construct an effective set of fine-grained feature outputs that could in turn be fed as inputs to a more specialized virtualized security function residing in a cloud environment that can be selected accordingly to fit the measures and strategies needed to mitigate the risk. And once in a warning state, the security function can be requested on demand so once it has been enforced, can be detached from the network (see Fig. 7).

Emulating the concept of unified services in conventional networks, in SDN a unified security framework (see Fig. 8) should be developed to guarantee security at all network levels, including all actors and layers involved in the SDN architecture. This means that security should begin from the early stages of network construction and applications development, passing through network deployment and services installation stages, monitoring, network state checking and policy enforcement stages and finally until enforcement of forensic and network

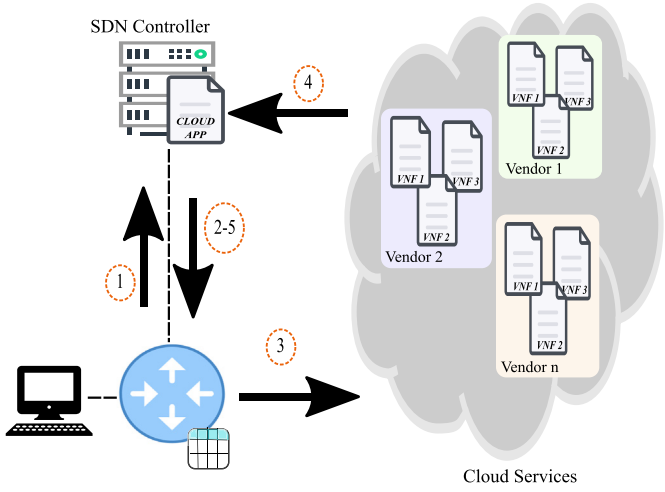

Fig. 7. On-demand request of cloud-based security functions: 1) A network device requests a security function; 2) The controller (or network application) instructs the device to request the security function from a cloud service; 3) Device forwards the request to the cloud service; 4) The virtualized security function instructs security actions to the controller (network application) and 5) The controller (network application) instructs flow-rule(s) that accomplish security actions.

resilience schemes. SDN-security development framework, including programming languages, debuggers and compilers, provides SDN developers with tools for the development of secure network applications and security policies. It also provides them with mechanisms to trace down vulnerabilities and software failures before application deployment, this guarantees that network applications are installed only after exhaustive sanity-checks, functional and coverage tests, and vulnerability assessments are accomplished.

On the one hand, security policies, network state monitoring schemes and attack detection systems guarantee network protection and reaction against suspicious network behavior and sudden unexpected changes. While on the other hand, encryption and ciphering schemes protect the data being exchanged within the network through all interfaces and communication channels, and besides, trust mechanisms ensure that only authorized and authenticated elements are included in the network at the different network layers.

Forensics and network resilience schemes ensure, in first place, that after an attack situation the network can keep in operation uninterruptedly and that proper network reconfigurations and attack mitigation processes are carried out to relieve from the effects caused by the attack and to restore network operation. And in second place, to trace down the attack sources, collect attack evidence, isolate devices and applications involved in the attack and then generate the reports and descriptions that would be later used for further analysis.

4. Classification and overview of SDN security solutions

In this section, we provide a comprehensive review of new security solutions to attack vectors and threat surfaces that target the SDN architecture. We present the proposals classified into eight categories that reflect either the main contribution to security or the main network concept that is embodied in the proposal: Attack detection, Virtualized/Cloud-based security, Threat and attack mitigation, Protected and secure sessions, Network state monitoring, Vulnerability assessment, Forensics and finally Integrated security framework.

4.1. Threat detection

The development of security applications for reliable traffic inspection and attack detection can be achieved leveraging the capacity of the SDN control plane to request flow features from infrastructure devices and then infer about network state, traffic patterns and other characteristics from colleted information. Software routines in specialized security applications are designed to detect anomalous behavior by extracting and analysing the network information requested by the control plane.

Software programmability and network flexibility help to enforce the replacement of fixed middlebox appliances in SDN networks. Software-based solutions and machine-learning approaches can be managed to deliver effective security schemes (see Fig. 9). For instance, Learn2Defend (Tantar et al., 2018) incorporates open-source software platforms and machine-learning capabilities to offer attack detection and mitigation by means of a system prototype that benefits from information regarding the network behavior and the current network state. The training set for the machine-learning engine is built on-line by labeling a set of ports and network protocols. Then, flows traversing labeled ports and using labeled protocols constitute input data for the training phase. Later, the construction of the detector instance is triggered once a fixed amount of training samples were added to the training data set. This detector is deployed as an application in the controller.

In (Le et al., 2015) the authors involve machine-learning operations in flow packet inspection to provide an intelligent SDN-IDPS solution. The design is basically an integration of a comprehensive set of machine-learning features that can be used to construct an effective framework for packet inspection and attack detection.

According to the patterns and singularities observed in the attack features, the classifier is capable of labeling the features from regular flows as normal or anomalous, therefore detecting flows issued by an attack source. Tests executed by the authors show that the IDPS solution exhibits an acceptable performance under heavy workload attack conditions.

The work presented in (Ajaeiya et al., 2017) exhibits two novel design premises for the deployment of Network-IDS (NIDS) in SDN. In the first place, an efficient perimeter defense scheme against external attacks is consolidated to reduce the set of switches that report network features to the IDS to only the edge switches. A second measure consists in letting edge switches forward run-time network statistics in fixed time-spans rather than mirroring all traffic flows every time. In addition, NIDS functions might not be processed by the controller but they should rather run in parallel in the same controller hardware appliance.

NIDS architecture developed in (Abubakar and Pranggono, 2017) benefits from the combination of two conventional approaches: in the one hand it leverages a signature-based approach that relies on the use of well-known security attack mitigation systems. In the other hand a machine-learning approach, built using a Back-propagation algorithm, acts as a pattern recognition system whose aim is to guarantee defense against unknown and hard to detect security threats.

Signature-based approach is implemented as a physical detection appliance installed at a fixed location in the topology. Meanwhile, the pattern-recognition system, that is built on top of the SDN controller, leverages flow statistics collected by the controller to detect anomalous traffic patterns and later instruct the SDN controller to install new flow entries in data plane devices.

Authors in (AlEroud and Alsmadi, 2017) highlight that traditional signature-based anomaly detectors exhibit considerable and non-tolerable error margins regarding detection of intrusions. Authors pointed out that such mechanisms can only make inferences upon features provided by OpenFlow header fields. To cooperate with this situation, they developed a graph-based intrusion detection scheme that in first place builds a robust database of attack flows, where each flow is aggregated using packets that share common network features and timestamps. Later, flows are fed to an IDS, this is done with the aim of labeling flows as either attack or benign flows depending on the concept emitted by the IDS. Labeled flows are then stored in an attack database.

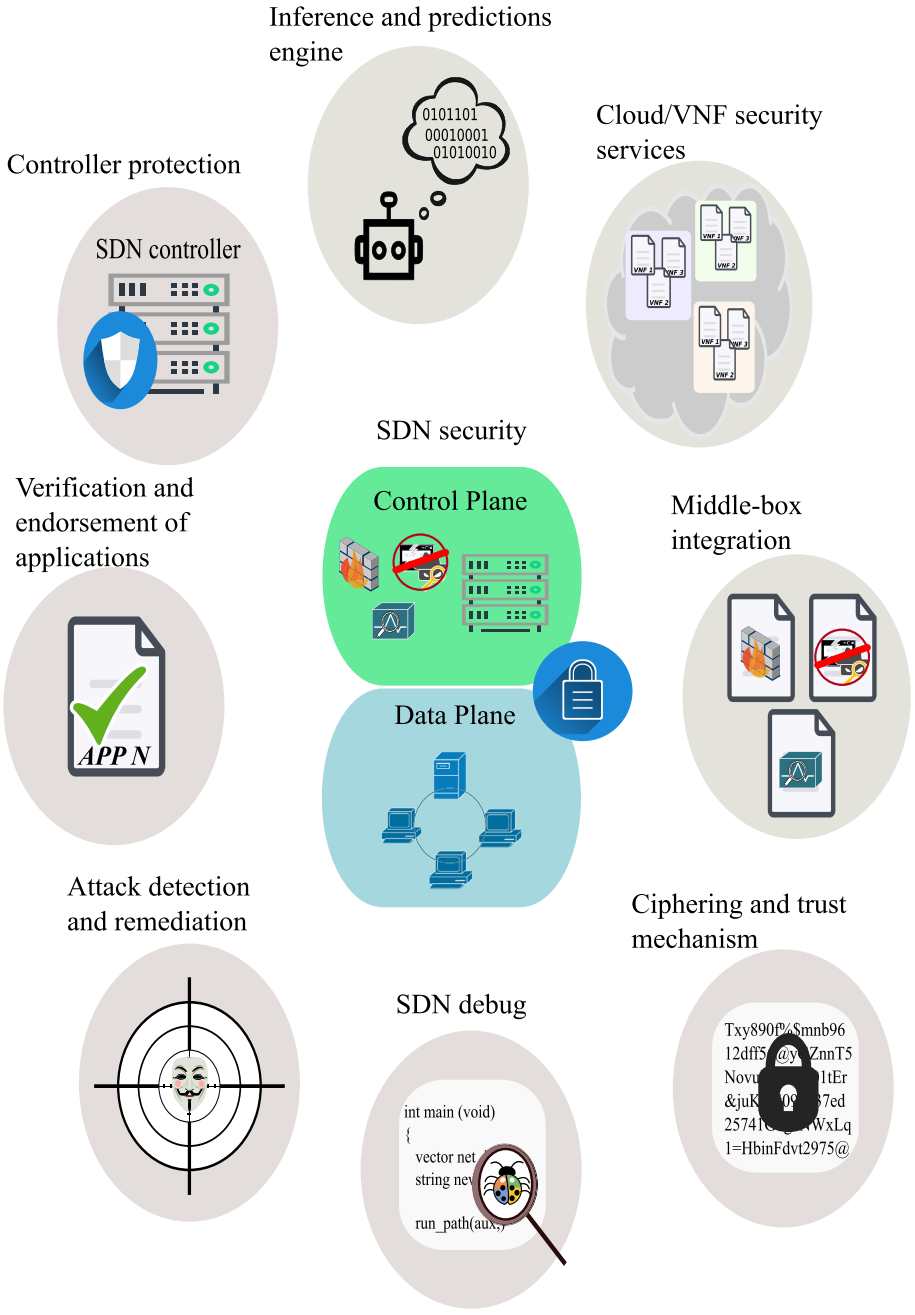

Fig. 8. Unified SDN security framework: This figure portrays the main components that should be embbeded in a pretended secure SDN architecture. Each component individually benefits certain security aspects though a more comprehensive security architecture should tend to foster synchronization of all components and unified management and supervision of the complete security scheme.

Every flow in the database is characterized by its label and its unique flow features. A flow monitor module is used to collect testing flows by requesting some flows to the switches. Those testing flows might be fed later to a classification engine that classifies flows either using a K-nearest neighbor algorithm or a Markovian approach.

Athena (Lee et al., 2017a) is a security framework that supports network administrators and security developers in the deployment of applications for anomaly detection in large scale networks without incurring in excessive development efforts; rather making development a seamless process. Athena framework is deployed on top of control instances in distributed SDN controllers.

The framework is built on a basis of 5 fundamental blocks: 1) a feature collection/generation block extracts relevant features (for instance, match fields, byte statistics) from control packets exchanged in the network. This block also stores the network features in a distributed database so they can be available for other modules within the framework; 2) an attack detector/manager block provides actions and machine-learning algorithms that can be used to construct user-defined detection models that leverage extracted network features in the detection of potential anomalies. Any computation required in detection tasks is delivered to a distributed computing cluster to relieve network devices from complex processing; 3) an attack reactor/manager block translates mitigation actions suggested by detector block into flow rules ready to be installed in data plane devices; 4) a resource manager and 5) UI manager to provide a friendly interface to interact with users and to display reports and results.

In summary, Athena offers an intuitive development model for the automatized construction of anomaly detectors. Developers can develop

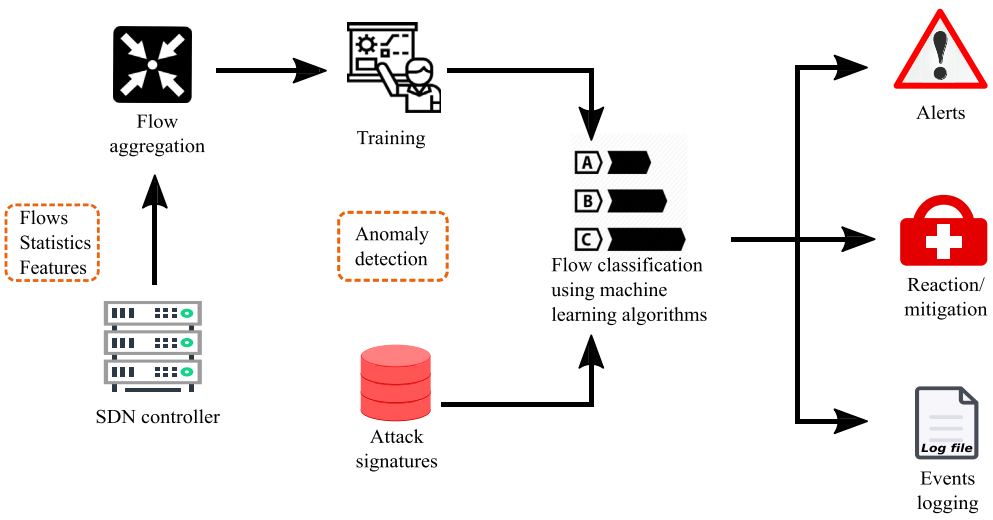

Fig. 9. The enforcement of supervised and unsupervised learning techniques (mainly classification and clustering) fosters the introduction of novel mechanisms that in general cooperate with real-time supervision of data exchange for detection of anomalies in network packets, northbound interfaces and control packets. Although this techniques strongly depend on data collection and selection, feature engineering, training and validation phases, the range of applications is constantly growing and in general, are a good fit in the SDN security context.

a strong detection engine following these simple steps: 1) request a set of network features, 2) generate a detection model enforcing a detection algorithm, 3) test the model against the features and 4) obtain measures and strategies from the reactor block.

4.2. NFV/cloud based security

Network functions virtualization and virtualized cloud services share a characteristic in common, both technologies use hypervisors to let different software solutions coexist in a shared hardware substrate, maximizing the utilization of resources provided by the hardware platform. In the former case a pool of virtualized services is locally available in the system and can be instantiated when needed, while in the latter case, the virtual functions are distributed in external systems and might be enforced by any network element with granted access to them. In SDN, a virtualized security environment enables the construction of a more sophisticated security system by enforcing the integration of multiple solutions provided by different vendors, and hence allowing an on-line construction of the security systems as virtualized resources that can be requested on-demand.

The enforcement of virtualized and distributed security instances contributes to enhance SDN performance since a reduction in the controller processing burden is exhibited when security functions processing is transferred to external computing entities (clouds and virtual service pools in switches). Likewise, the use of virtual services fosters innovation as SDN developers focus on the design of network interfaces to the virtual entities, while vendors of security services focus on developing security solutions.

Current firewall instances in SDN networks leverage from one of these two approaches, on the one hand, the controller-centric approach where firewall functions are designed as an application running on the controller, and, therefore, forward, deny and drop functions can be instructed to data plane switches in the form of flow rules. On the other hand, the NFV-centric approach where firewall applications are deployed as virtualized network functions in distributed cloud environments.

Authors in (Lorenz et al., 2017) identified some drawbacks in both approaches and proposed a solution to overcome their shortcomings with a hybrid stateful firewall function that leverages on the best features from both approaches. The proposed solution provides a scheme that is similar to an NFV-centric approach with the firewall application instantiated as a virtualized function running in the cloud. The difference resides in the implementation of an additional offloading optimization algorithm that instructs data plane devices to safely forward data packets that belong to already tracked (analyzed by firewall) flows without further intervention of the virtualized firewall instance. This way, NFV-firewall application can be relieved from inspecting every single network packet. This proposal can benefit scalability, throughput and helps in reduction of packet latency.

In the context of Industrial Networking Systems (INS) authors in (Cheminod et al., 2017) present a proactive security data plane approach by the introduction of Active SDN switches, A-switches. These active switches are provided with an additional virtualization layer that allows the instantiation of virtualized security functions in the form of individual security applications or even more robust security service chains composed by many specialized virtual functions. The SDN controller dictates the activation of virtualized security functions to the switches through a load manager module that resides in A-switches virtual layer. A-switches supress the need of middlebox appliances at fixed locations enabling a scalable and flexible approach for SDN security in industrial environments as they let security functions to be directly deployed in forwarding devices, and they help in fine-grained distribution of specific security functions in sensitive network segments.

The security architecture for SDN presented in (Lee and Kim, 2017) translates security policies into comprehensive virtualized security services that are built mapping virtualized security functions to combinations of four specific policy functions, namely: 1) Separating; 2) Chaining; 3) Merging and 4) Reordering.

Based on the attack itself and the attack conditions, a security policy meta-model constructs a security system as a combination of the aforementioned four policy functions that is then translated into tangible virtual network functions using the service chaining engine in the controller that later instructs service chaining pool in data plane devices assigning certain security roles and then installing the virtual network functions that accomplish the security action against the attack flows.

The work in (Hyun et al., 2018) demonstrated that network security functions deployed in cloud environments can be integrated along with SDN basic filtering functionalities to enable a comprehensive security architecture. In first place, the proposal introduces the development of a set of standardized interfaces for the translation of high-level security policies into a seamless selection and enforcement process of heterogeneous network security functions provided by multiple vendors that are available in cloud environments. The remaining of the proposal focuses on providing integration between network security functions and data plane packet filtering aiming at the deployment of a dynamic packet inspection and attack detection infrastructure where network security policies can decide if packets from a specific flow need a heavyweight treatment by specialized network security functions in the cloud or otherwise the SDN controller is instructed to issue pertinent flow rules to treat such packets directly at the data plane.

This architecture can benefit network security by allowing multiple security functions (firewalls, deep inspection appliances, payload checkers, etc.) to act on a single flow packet in a service chain fashion. It might also benefit network performance as it delegates filtering functions to data plane devices thus avoiding all packets from being forwarded to the cloud.

The security framework presented in (Lin et al., 2017) collaborates to reduce control processing overhead by the fact that security tasks are performed in a security instance physically decoupled from the network controller. The framework exhibits modularity and reusability given that security functions can be embedded as virtualized functions in data plane appliances. The framework design can be summarized as a collaboration between a security meta-functions database and a security service orchestration center provides a mechanism for the composition of security services by means of the translation of high-level security service requirements into virtualized security functions. Then, the virtualized functions are deployed on security agents located in the data plane.

4.3. Attack remediation

Since most security threats commonly observed in conventional networks can be reproduced in SDN scenarios, there should be the possibility to build threat mitigation strategies enforcing the constitutive characteristic of the SDN architecture, for instance network programmability and centralized flow management. A variety of solutions may be devised in the form of control applications since all information concerning network state is available for the controllers, then in response to any deviant behavior the application can instruct the controller to issue the pertinent flow entries that should mitigate the situation.

For instance, a robust SDN attack-mitigation application might be comprised of a detection engine that is capable of distinguishing the specific characteristics of an anomalous network behavior according to certain unique features described in a set of security policies, and a reactor module that directly enforces a set of mitigation actions in the compromised network elements (flows, switches, channels, network interfaces), according to the features described in the policy.

Next, we show some SDN-based mitigation approaches for different security attacks.

4.3.1. Flooding/denial of service attacks

In (Sahay et al., 2017) the authors propose a mechanism to mitigate link (path) congestion attacks (flooding, DoS, DDoS, etc.) in carrier-grade networks by offering an intuitive policy management and enforcement framework in which high-level network policies stored in a repository can be translated into newly calculated paths that can be used to redistribute attack flows and alleviate link congestion. Following the detection of an anomaly, customers may raise an alert to a network monitoring component attached to the ISP (Internet Service Provider) controller, then the monitoring component can extract concerning information from different paths to infer about the congested paths.

A policy decision point gathers the features collected by the monitoring component and requests policies from a policy database to decide what actions should be taken against the attack flows. Then, a policy orchestration and implementation module uses flow information and the decision actions described in the policy to elaborate the computation of new forwarding paths and to issue the flow rules (forward, drop) to the switches in the network to redistribute congestion flows in the newly computed uncongested flow paths.

DDoS attacks launched using vast low-traffic flows are hard to detect. Most traditional detection engines fail to detect such attacks, and often report non-negligible rates of false positive and false negative detections. Authors in (Dong et al., 2016) present a solution that takes advantage of Sequential Probability Ratio Test (SPRT) to provide a scheme that can detect distributed vast low-traffic packet-in flooding attacks in very few rounds of flow analysis, this scheme can successfully identify compromised network interfaces participating in the attack, and authors claim that this proposal reports zero false positive and false negative error rates.

4.3.2. Side-channel attacks

The proposal in (Conti et al., 2018) is a security countermeasure against the Know-Your-Enemy (KYE) attack (Conti et al., 2017) which is a side-channel attack that exploits the on-demand flow rule installation behavior exhibited by reactive Openflow-based SDNs. In general, the KYE attack is divided in two phases, the first one corresponds to the probing phase in which the attacker sends probe packets to the target switch and observes the flow rules installed in reaction to the probe packets. In the second phase, the inference phase, according to the information gathered in the prior phase the attacker is able to discover information about the network configuration, the network policy enforced for specific traffic flows and even the exact flow entries issued for certain flows.

Flow obfuscation is the security countermeasure suggested against the KYE attack. This countermeasure exploits the ability of Openflow switches to modify packets in transit. When the attacker sends a probe flow to a compromised switch, the controller issues a single flow rule that instructs the switch to modify some flow header fields and to forward the packets to another switch, the process is repeated for a pre-established number of switches. When the flow reaches the last switch then the controller issues the corresponding flow rule that enforces a security policy for the original attack probe flow. The set of switches involved in flow obfuscation process are said to belong to what is called the obfuscation path.

Flow table overflow attacks were foreseen to happen in SDN networks from the fact that flow rules have to be stored in physical finite size memory locations in switch hardware. Authors in (Leng et al., 2018) performed several attacks targeting a switch's flow table with the aim of identifying some special property that attackers could leverage to infer about the exact moment a flow table is completely full. The authors came up to the conclusion that the highest RTT (round-trip-time) values can be observed in the situation when a flow does not match any installed rule and the flow table is full as well.

Once with flow table overflow attack fingerprint properly identified the authors provided two defense strategies that can be deployed either individually or joint. In first place, a flow rule compression mechanism can be delivered using routing aggregation algorithms along with a packing optimization strategy. Second, to avoid RTT inference a multi-flow table architecture can be used to extend flow table capacity and prevent overflows, in this approach flow tables could be installed in free RAM memory slots once TCAM is completely full.

4.3.3. Rogue device infiltration

Network Flow Guard (NFG) (Cox et al., 2017) is a solution designed to defend SDN networks against attacks infringed by illegal intruders that infiltrate private networks taking advantage of unsecure access points intentionally or unintentionally deployed inside network topology. NFG detects rogue access points relying on a passive packet inspection scheme combined along with an active detection engine.

NFG's passive inspection scheme includes three filtering stages: In the first one, a suspect device is matched against a pre-defined "whitelist" of devices (NAT, wireless routers and wireless switches) allowed to be connected to the network. The second step uses an algorithm to extract TTL (time-to-live) values from packet headers and then it looks after decremented values to track the existence of unauthorized access points in the network. In a third stage, supervision over switch ports allows to detect the amount of MAC and IP addresses associated to a single port, and in case the count exceeds one, then the port is flagged and the switch is isolated from the network. The final step

in the passive scheme is in charge of recording average TTA (time-to-acknowledge) values in TCP SYN handshake messages at every network switch port, a global average called port average is computed and if any outlier with respect to port average is observed at any port, then that port is flagged and its traffic is redirected to the active detection engine for deeper testing and isolating rogue devices from the network.

4.3.4. Malformed packet injection

PacketChecker (Deng et al., 2018) is a mitigation measure against security attacks derived from malicious packet injection in SDN. Using conventional hash data structures and aiming at tracking the legitimacy of network packets, PacketChecker creates mapping instances to bind unique and incorruptible details that characterize data plane devices, for instance MAC addresses, DPIDs and switch ports.

In first place, PacketChecker gleans header fields information from the very first Packet_In message received from every single switch connected to the network, this is done to subscribe the switch and to record corresponding hash maps in storage tables. Next Packet_In messages must be verified against recorded information so that packets exhibiting any incongruence are immediately dropped to prevent other controller applications from parsing them. The controller is instructed to issue corresponding flow rules to drop packets that out come from the identified injection sources. PacketChecker exhibits a seamless performance as it does not interfere with parameters of the Openflow channel like workload and bandwidth.

4.4. Identity and access management

In SDN, high sensitive information is conveyed through the control channel, that is why many attacks focus on eavesdropping on that interface to get enough information to compromise the network. Enforcing strong encryption mechanism on the communication channels enhances security since although attackers might still be able to leak ciphered data they might not obtain real data in plain text (see Fig. 10). However, clever attackers might have the expertise to infiltrate the network using a compromised or rogue network element, in that case data protection demands additional security measures to ensure that outsider network agents are not allowed to participate in data exchange. To prevent any outsider from joining the network a trust mechanism for authentication and authorization of network privileges should be enforced. In this way not only data is protected using ciphering schemes but unauthorized access to the network is prevented as well.

Based on the fact that the encryption mechanisms offered by the Openflow stack for control channel protection are optional and hard to deploy, the authors in (Gray et al., 2017) proposed a novel mechanism for network device fingerprinting that provides an accurate authentication strategy. The authors claim that though coming from a common vendor, network switches often exhibit different implementation features and expose observable discrepancies regarding processing times in the execution of the same functionalities, these features constitute a device fingerprint.

All devices recently attached to the network got to be registered at the fingerprinting module before they are put into operation. Then, network operators might trigger the fingerprint construction and each time a device connects to the controller the fingerprinting module blocks any network application attempting to establish a link with the new device. This blocked state continues until the authorization process is finished and if the authentication process is successful the block is removed, otherwise such device keeps isolated from network operation. Upon authentication failure, the fingerprinting module alerts network operators that a potential rogue device is being attached to the network.

In (Ranjbar et al., 2016) authors leverage a centralized SDN control scheme to arbitrate the establishment of secure sessions. Using a mechanism that checks credentials and parameters in clear-text format that are exchanged during the handshake phase of a secure session. The mechanism is able to validate items like the strength of the encryption

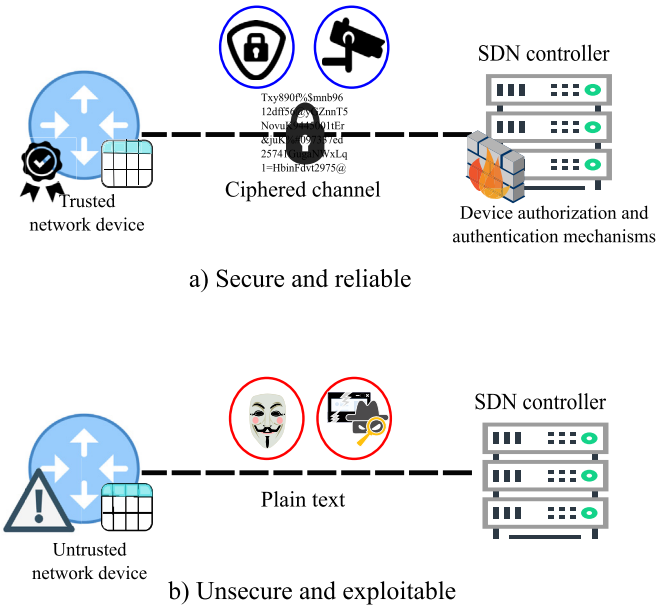

Fig. 10. Control channel protection: The separation of control and data planes exposes a single point of failure for the entire SDN. Then, implemenation of specialized security measurements in the control interface is mandatory in any SDN deployment since a poorly secured control channel might be exploited to compromise the SDN controller, the infrastructure devices and eavesdrop on sensitive flow data regarding network state and configuration.

algorithms as well as both the expiring dates and issuers of the security certificates used in the handshake exchange. Such features are validated against a set of constraints previously described in a pre-defined set of security policies.

The aforementioned security policies include a wide set of user defined security constraints, for instance: the TLS protocol version permitted for secure channel sessions in the network, or even the list of trusted certificate issuers whose certificates are valid in the security negotiations that may take place in the network. The security policies are fed to a policy-box installed on top of SDN controller. Then, features present in the handshake messages are matched against the constraints described in the policies. When the handshake phase is completely verified, the controller proceeds to install flow rules in the switches so a reliable path can be established between the devices that will perform as the ends in the data exchange.

The lightweight mechanism for secure access control in SDN presented in (Mattos and Duarte, 2016) prevents unauthorized access to network resources by enforcing the existing IEEE802.1x standard and the EAP (Extensible Authentication Protocol) along with RADIUS server implementation. The design includes an application for device authentication that is deployed to instruct forwarding/discarding rules to the controller according to results of an authentication processes that takes place at the RADIUS server.

KISS (Kreutz et al., 2018) framework proposal is intended to overcome drawbacks of TLS implementation by offering a mechanism for device registration and association that enforces an encrypted secure channel, and a lightweight and decentralized random key generator/verifier that can be adapted to work at a per-message basis. The encrypted secure channel uses the NaCl cryptographic library along with Poly 1305, SHA512 MAC and some other renowned hashing schemes. Random generation of encryption keys is carried out by the Integrated Device Verification Value (iDVV) mechanism that the authors claim can provide the system with a reliable randomness in key generation.

A key distribution center is adapted for the execution of registration and association processes. In the registration stage, devices can register

and obtain secure keys that will be used in the association process prior to the establishment of a communication channel. While in the association stage both communication ends, initiator and receptor, request a seed-key pair for the synchronous generation of secure channel crypto keys using the iDVV mechanism. The crypto keys generated might be used to cipher/decipher the communication channel. As claimed by the authors, the deployment of a lightweight crypto mechanism results in low latency in key generation process compared to other well-known crypto schemes.

SM-ONOS (Yoon et al., 2017b) is an administrative permission system that manages the access levels to the controller, and even to other network resources, that is requested by SDN third-party applications. The SM-ONOS authorization system relies on the enforcement of two types of access control policies for network applications: the first type consists of a set of access level policies defined by application developers in which different roles can be requested to grant certain access privileges to network resources and services that reside in the application-control interface. The second type of access policies refer to the constraints fixed manually by network administrators to third-party applications. Such policies describe the type of flows that third-party applications are allowed to treat and the network segments they are permitted to act over. A network application cannot be activated until both types of policies are totally approved by network administrators.

4.5. Network state monitoring and analysis

Monitoring network activity might be thought of as a first step in the introduction of wider security implementations. If a security framework can maintain a continuous global view of current network state then the reactions to unexpected situations can be expedited when certain measures have been previously worked out using the features identified in the monitoring phase. One revealing characteristic often found in different attacks is the abrupt change in network conditions and activity that just occurs when the attack starts. A complete monitoring solution might identify such sudden state violations, and then feed enough and appropriate input to a detection engine or otherwise directly trigger a security application that contains the proper reaction to mitigate the attack.

Path Class Approach (PCA) (Wrona et al., 2016) is a data-centric security approach in which flow packets, referred as data objects, are mapped to certain forwarding paths using certain labels. The selected path is supposed to guarantee some specific essential protection requirements and certain set of security policies that are meant to assure confidentiality, integrity and availability of the information contained in those flows. The PCA evaluation testbed presented in (Wrona et al., 2017) implements a comprehensive software and hardware structure that is supposed to dynamically estimate link availability based on current link usage. This link estimation can help to program packet routes that meet data-centric security requirements for specific data objects.

Authors claim that in practical testing scenarios, the PCA mechanism exhibited effective reactions to path congestion generated by DoS attacks; the mechanism efficiently removed congested links from the set of existing flow paths and replaced them with new paths that enforced data-centric protection requirements and security policies.

In the context of defense to attacks in SDN, an effective security framework may be built from the integration of specialized individual security components developed for the accomplishment of security tasks that are away from those that are focused on triggering an attack mitigation scheme. That is the case of Global Flow Table (GFT) (Qiu et al., 2017), a security approach devised to provide other security applications with relevant details about network flows in a way that third-party SDN security applications can use such details as resource for the benefit of their own performance.

GFT builds a global flow table database by requesting flow tables information to a limited set of network switches, it also computes every flow path in the network using algorithms for agile path computation. These features provide a global view of the state of all network flow paths. GFT is supposed to collect all flow tables from all switches in the network, though it can be a resource and time consuming task.

To highlight their contributions in, the authors in (Qiu et al., 2017) developed a set of security applications that leverage the information provided by GFT. For instance, a man in the middle application automatically detects this attack by checking the global database and identifying flows whose source and destiny MAC addresses are the same.

An additional security plane that is able to cooperate with data and control planes in security enforcement is devised in (Hussein et al., 2016). This novel SDN secure layer is designed to offload the controller a significant amount of security processing overhead. The security plane includes security agents distributed in data plane devices and a centralized detection engine that works in parallel with the controller sharing the same hardware substrate.

The performance of the security plane is summarized as follows: the security agents capture packets not matching certain mapping features, then packets are forwarded to the centralized security module which collects related information from all network switches and enforces security policies. Once a detection occurs a special command is sent to the controller via REST API, this command instructs the controller to install blocking rules for specific IP addresses in all switches or in individual ports per switch.

Sphinx (Dhawan et al., 2015) offers a security scheme that protects the network from topological and forwarding state violations. Sphinx extracts meta-data from the Openflow packets exchanged between the controller and data plane devices in order to build incremental flow-graphs to maintain a global view of both the network topology and the current forwarding paths. The framework performs a run-time defense against anomalies by checking and validating meta-data extracted from flows. Such data is matched against both a set of administrative policies and a set of historical features that describe the historical network sate and flows forwarding behavior.

Sphinx is built as a seamless layer between control and data planes, this condition allows easy collection and parsing of Openflow packets, meanwhile, relieving the controller and data plane devices from executing additional processing needed to accomplish security tasks.

4.6. Security assessment

Vast amounts of network security risks and issues can be attributed to the following situations: bad configurations; wrong implementations; non-provisioned or undetermined reactions to unexpected input or run-time situations; wrong protocol specifications and designs; etc. It is clear that preceding the release of a production network demanding tests and assessments have to be executed to guarantee both the functionality and the security of the network. Unfortunately, many security tests are not so strict and just check some common features, while leaving untouched hidden security problems that might not be visible or evidenced until certain special conditions are met, anyhow such unchecked problems remain as a dormant threat that a clever deviant user might abuse in the future.

For the reason explained above is that it is a compelling task to develop security solutions that provide an exhaustive test for uncovering vulnerabilities. A complete secure network deployment should at least guarantee that the network is not vulnerable and is properly shielded to attack vectors derived from any of the situations mentioned above in the former paragraph. In SDN such kind of solution is achievable by means of security applications that can perform a comprehensive vulnerability assessment either on-demand or periodically.

Delta (Lee et al., 2017b) is a proactive approach used to uncover attack vectors and vulnerabilities in the Openflow stack by taking advantage of fuzzing techniques that reveal security failures derived from errors that occur after handling unexpected input or data at

the different interfaces that bind any pair of layers in the SDN stack, namely: Application-Control, Control-data plane, and Host-data plane interfaces.

The system architecture is composed of a security agent manager and security agents distributed at each interface level. The agents are provided with specific modules to replay attack scenarios, to apply one of two available fuzzing (Oehlert, 2005) testing techniques, and with an interface to receive management instructions from an agent manager. The agent manager is located at the controller, it manages the different security agents and gathers testing results obtained when an attack scenario was successfully replayed.

The authors defined a seven criteria assessment (controller crash, application crash, internal-storage poisoning, switch disconnection, switch-performance downgrade, inter-host communication disconnection and error packet generation) to expose vulnerabilities found in fuzzing tests. That is, if any fuzzing test scenario results in the occurrence of at least one situation described in the assessment, then a vulnerability related to the test inputs is considered to be present in SDN architecture.

Using Delta system architecture, the authors reported the discovery of new SDN vulnerabilities as well as the SDN networks being prone to well-known networks attack scenarios that happen in conventional networks.

4.7. Forensics

Forensics play an important role in the reconstruction of cybercrime scenes and in the recording of valid datasets that might be used to track down and uncover the source of certain security threats. SDN Forensics framework (Zhang et al., 2017) is one among a few forensics proposals in SDN and consists of a multi-layered approach that enables functions at every SDN stack layer for: treatment of evidence data; anomaly detection leveraging artificial intelligence, alarm triggering; and data formatting instances for the evidence collected in the attack scene.

A data acquisition and extraction instance is able to construct evidence datasets using information collected from control and run-time logs, memory images and physical data storage (disks). Then, a data fusion instance integrates in the form of clusters all the information extracted from physical and software sources. The evidence clusters become the input source that is fed for data analysis using machine-learning techniques that specialize in anomaly detection, triggering alarms, scene reconstruction, and documentation of the reported attack.

4.8. Integrated SDN security framework

Section 3 highlighted the need for a wider SDN security solution conceived to cooperate with the materialization of a unified security architecture for SDN. The architecture should pose a holistic security solution able to offer a fine-grained control of all security aspects in the network, while at the same time plays the role of a composition of single components that individually attend specific security issues. The building blocks of the architecture should offer mechanisms and schemes focused on the enhancement and protection of the entire system components and interfaces rather than just offering solutions to punctual attack vectors or anomalous situations. Such feature might represent an extra contribution to overall security enforcement by the materialization of implicit proactive security measures since a wider solution might inherently offer supervision and control for multiple attack surfaces that coexist in the single system component or interface.

In addition, the security architecture should focus on attending high-level features found in security issues instead of dealing with low-level details. This might guarantee that the security solution acts on a wider range of scenarios and situations while providing sufficient and effective response to specific security incidents. For example, an architecture-oriented solution for the protection of the control channel should be devised as a mechanism that is in general able to guarantee security conditions like confidentiality, integrity, availability of data without concerning about covering details related to specific attacks that compromise such security constraints.

Conflicting flow rules can force an unintended flow-entry replacement process in the flow tables of SDN data plane devices, such phenomenon can be attributed to malicious network applications tampering with network controller to instruct fraudulent or misguided packet forwarding behavior. To defend against the aforementioned situation and to tackle network attacks, authors in (Hu et al., 2015) propose a security system architecture framework that leverages on three main aspects: 1) fine-grained naming scheme for flow entries; 2) packet data scan for threat detection and 3) an authorization scheme for network applications.

Regarding fine-grained naming scheme of flow entries, this proposal suggests the utilization of a role for the individualization of different policy creators, and integrated with a security level classifier for every role. The combination of different roles and security levels allows the specification of appropriate access rights and permissions that can be given to a specific application in order to grant its access to the controller. The main task assigned to the controller is to receive policies from different applications and translate them into flow entries according to: the characteristics of the policy itself; the role assigned to the policy creator; and the security privilege level of the application.

Regarding packet scan and threat detection, the framework includes amendments to Openflow match fields by adding an additional field to the Openflow protocol standard to specify if certain flow packets should be forwarded to an external security appliance for a deeper packet inspection to detect if the packet belongs to certain attack data flows.

Regarding the authorization scheme for network applications, the deployment of any accepted and widely-spread authorization scheme for network applications is suggested to prevent the installation of flow entries issued by fake and fraudulent applications. In consequence, applications that fail to correctly authenticate with the network controller cannot be granted with permissions for flow rule installation.

Overall system framework is basically composed of: an automated security management function that helps the SDN controller in detection and mitigation of attacks; and a network security monitor function that assists network administrators in monitoring the network status.

To accomplish all aforementioned design and tasks the SDN controller is conceived to account additional specialized modules embedded in its system structure, namely: a module for policy input; a module for mutual authorization and authentication between the different devices that are distributed in the SDN architecture; a network security monitor module to create and display log files and reports; a module to provide encryption and cryptographic functions; and a system security management module that provides functions for control and synchronization of the entire system.

NOSArmor (Jo et al., 2018) is a proposal that builds a complete SDN security framework leveraging the integration of several individual SDN security approaches. The different security approaches are integrated into the framework as security building blocks (SBB) that separately address specific attack vectors and security risks that threaten the network controller. Eight SBBs conform the security framework, the following five out of the eight enable security measures against attack vectors originated at the application layer: role-based authorization, process separation, rule conflict mediator, resource manager, and system call checker. While the other three SBBs are supposed to mitigate risks originated at the data plane: Openflow protocol verifier, link verifier, and hots-location tracker.

The selection of the SBBs was made in order to address the following security concerns: 1) unauthorized applications accessing the internal storage or enforcing switch commands; 2) verification of trusted hosts; 3) crafted protocol messages from malicious hosts that are issued to poison the controller network information; 4) applications installing conflicting policy rules into the controller; 5) malicious applications

misusing the controller core system functionalities; 6) rogue devices issuing malformed Openflow messages and 7) applications abusing network resources.

To the best of our knowledge this security framework is the only one of its kind that we have reviewed in current SDN security state-of-the-art; most of the proposals in this article either focus on giving a security solution to a single SDN attack vector or try to build a holistic SDN security solution leveraging different software techniques and the most relevant features of SDN, for instance, network programmability. Conversely, this approach leverages a widely diffused concept that is the integration of different specific solutions to materialize a new wider solution that serves many security purposes at the same time enforcing the best elements and features from prior works. The main concern regarding this approach is the introduction of additional processing load to the controller performance which might result in increasing latency and decreasing throughput, although authors claim that performance is not significantly affected and that performance metrics are not different to those exhibited by well-known state-of-the-art SDN controllers.

Tables 2 and 3 present a taxonomy that in the first place divides all proposals in two main groups: the first one corresponds to the works that leverage SDN to enable certain security mechanism or measurement, while the second one makes reference to the proposals devised to provide a secure SDN architecture. Then the proposals are subdivided in the categories presented in this section: Threat detection; NFV/Cloud based security; Attack remediation; Identity and access management; Network state monitoring and analysis; Security assessment and Forensics and Integrated SDN security framework. Such categories correspond to the most relevant security feature/contribution that reflects in each approach either through the design, implementation, deployment or even the results presented by the authors.

5. Discussion

5.1. Faulty system integration and system complexity

Security is as strong as the weakest link, any loose end in a security approach exposes the entire network system to be compromised sooner or later. For that latter reason it is suggested to run extensive assessments and tests for the identification of critical points that could be leveraged to bypass the security measures and then compromise the network elements under threat. Besides, implementation of a comprehensive security system implies adding complexity to the network and reducing its usability, the more secure a system is, then more complex functions and interactions are needed to keep it working properly, and in consequence the more difficult for users to operate it in an easy and intuitive way. The former and latter aspects that have been depicted strongly influence in the level of protection that any security strategy can offer to the system it is intended. For instance, a faulty or incomplete solution cooperates to mitigate the issue that it was designed for but that does not ensure that any other vulnerability has not been triggered as a consequence of the alterations to the system state induced by the integration of the security scheme to the system itself. On the other hand, a non-user friendly security scheme could result in users deactivating it or bypassing the configurations needed to guarantee that the scheme effectively protects the system, thus leaving the system with incomplete protection and open to be compromised if any uncovered misconfiguration exposes a weak element in the system. For instance, users bypassing the update of network policies and evading registration, authentication/authorization schemes when integrating new elements to the system.

5.2. Inconsistencies in the design and implementation of security strategies for SDN

In order to contextualize, an inconsistency can be considered: 1) a flaw in either the design of the security scheme or in its implementation; 2) a misplaced idea not fitting with the current system architecture and configuration or 3) a sensitive or critical detail that has been ignored and may affect either or both the correct implementation of the security scheme or the results expected after deployment of the security strategy.

Inconsistencies in security proposals: Substantial changes to network definitions and configurations: Proposals that suggest to include extensive changes and modifications to existing protocols and specifications. For instance, changes to OpenFlow specification to include additional configurations, features, protocol fields, data structures, etc.

Additional attack targets and vectors: Including additional hardware appliances results in an increase of the list of potentially exploitable network instances and attack surfaces. Not restricted to only network devices but to new interfaces and channels that can be leveraged if not correctly secured and protected. For instance, network performance is degraded as a consequence of the delay introduced by the transactions between network devices and the security services embedded on additional hardware appliances.

Vulnerabilities in virtualized services: It is not correct to assume that NFVs and Cloud services are completely reliable and vulnerability-

Table 2
Taxonomy: Proposals for network security leveraging SDN features.

Category	Proposal	Description	Contribution
Threat detection	Tantar et al. (2018)	Anomalous flows detected by observing outliers in historical flow behavior	Anomaly detector; statistical detector
	Le et al. (2015)	Flow classifier trained with information available in third-party security libraries.	Anomaly detector; ML-based detector
	Ajaeiya et al. (2017)	Flow classifier trained with known attacks' signatures.	Anomaly detector; ML-based detector
	Abubakar and Pranggono (2017)	Anomaly detection by combining neural networks and a signature-based IDS.	Anomaly detector; ML-based detector
	AlEroud and Alsmadi (2017)	Flow classification using markovian approach.	Anomaly detector; stochastic detector
Attack remediation	Sahay et al. (2017)	Mitigation of path congestion by flow re-distribution	Fair distribution of network flows
	Dong et al. (2016)	Detection of DDoS attacks	Anomaly detector; statistical detector
	Conti et al. (2018)	Remediation of KYE side-channel attacks	Attack detector
	Leng et al. (2018)	Prevention of flow table overflow attacks	Proactive attack prevention
	Cox et al. (2017)	Detection and neutralization of rogue devices	Rogue data plane devices detector
	Deng et al. (2018)	Detection of crafted malformed control packets	Anomaly detector
Network state monitoring and analysis	Wrona et al. (2017)	Distribution of traffic flows according to flow security requirements	Flow-path arbitrator
	Qiu et al. (2017)	Database containing all network flow tables and flowpaths	Centralized security monitoring
	Hussein et al. (2016)	Policy-based security layer for network monitorization and attack detection	Distributed security analysis
	Dhawan et al. (2015)	Network behavior and state monitorization	Stochastical detector

Table 3
Taxonomy: Proposals for secure SDNs.

Category	Proposal	Description	Contribution
Threat detection	Lee et al. (2017a)	Composition of security systems with off-the-shelf security resources	Language-based tool; security framework
NFV/Cloud based security	Lorenz et al. (2017)	Optimized NFV-centric stateful firewall	Virtualized firewall
	Cheminod et al. (2017)	Switches with virtualization capability	Virtualized security functions
	Lee and Kim (2017)	Security policies translated into virtual service chains	Virtualized security functions
	Hyun et al. (2018)	Interface to cloud multi-vendor third party security	Standard interface to cloud NFVs
	Lin et al. (2017)	Translation of security requirements into NFVs	Orchestration center for security NFVs
Identity and access management	Gray et al. (2017)	Fingerprint-based switch authentication	Authentication and verification mechanism
	Ranjbar et al. (2016)	Policy-constrained establishment of network sessions	Secure sessions arbitrator
	Mattos and Duarte (2016)	RADIUS-based authenticaton in SDN	Authentication and verification mechanism
	Kreutz et al. (2018)	Cryto-keys distribution for channel encription	Authentication and trust mechanism
	Yoon et al. (2017b)	Policy-constrained access of network applications	Access protection mechanism
Security assessment	Lee et al. (2017b)	Vulnerability assessment using fuzzing techniques	Detector of vulnerabilities
Forensics	Zhang et al. (2017)	Insights for the development of a multi-layered SDN forensics framework	Proposal of SND forensics framework
Integrated SDN security framework	Hu et al. (2015)	Security embedded in OpenFlow protocol headers	Generic SDN security architecture
	Jo et al. (2018)	Different security strategies as building blocks in a single framework	Integrated SDN security system

free. Excessive confidence in the security of virtualized services without deploying any protection measure often results in adversaries taking advantage of unprotected interfaces and misconfigured protection schemes.

Issues when leveraging Machine Learning approaches: Current machine learning approaches rely on the existence of appropriate training datasets and cannot deal with unknown attacks and vulnerabilities. Machine learning approaches exhibit a significant false positive rate that might not be tolerable under certain circumstances. Additionally, training phases in machine learning impose additional consumption of computational resources, without mentioning that attack datasets need to be constantly updated, a task that might not be automated in many cases.

Relying on human intervention and expertise: Security policies are prepared and updated by network operators that in many cases are not aware of the network state, it is, applications and devices deployed in control and data planes, and the issues and risks that threaten the network. Human intervention can either innocently or maliciously force misconfigurations in the security systems and perform internal jobs that result in network compromise.

Facilitating side-channel attacks: When security strategies include the utilization of an external system, for instance; a cloud server or a security appliance, a pattern in network behavior is exposed when network devices request services from such external entities. Clever attackers might infer about behavior patterns and network features by analyzing network behavior and state, then inferred features and information can be leveraged to develop mechanisms to bypass network security.

Standardization: Security proposals designed to work under specific constraints, namely: network controller, southbound protocol, switch specification, network topology, among others, might not be useful if any design constraint is not met at deployment, leaving the network prone to be compromised. For instance; security solutions designed exclusively for OpenFlow-based SDNs. On the other hand, standardized interfaces can greatly benefit the integration of different security systems when such systems are used as building blocks for the construction of a wider security architecture; such standardized interfaces ease the information sharing process between different solutions, therefore resulting in the inclusion of seamless and secure mechanisms to translate one system output into the adequate input for a more specialized system, for example, when passing the output obtained from a detection engine module as input to a reaction/mitigation system module.

Forensics: A network attack can be traced-down and all attack stages can be re-constructed leveraging the “attack marks” that can be gathered using, among others, packet captures, run-time logs and attack detection logs. Current SDN security solutions focus on attack prevention, detection and reaction/mitigation, while a complete and wide security solution should provide with enough tools to uncover the root causes of network attacks. From a narrow point of view, a forensic solution might pose as a reactive measure to security incidents but it is clear that all attack features identified after forensic analyses can be used as source for: the construction of additional security measures, identification of unknown attacks and attack vectors, augmentation of the size of attack databases (used by machine learning approaches), aggregation of attack flows for training detection engines; which from a certain perspective fosters a proactive behavior in the development of new strategies aimed to benefit network security and protection.

6. Open challenges in SDN security

6.1. Lack of standardized interfaces for the integration of NFV and cloud services in SDN security

Virtualized functions and services offer many new possibilities for the enhancement of security in SDN environments, as detailed in previous sections virtualized functions can be enforced by SDN to deploy security services directly on infrastructure devices or in the control plane, removing the utilization of specialized middleboxes and by the way offloading processing burden from control plane instances. Besides, by enforcing virtualized functions and services available in cloud environments SDN security benefits by enabling the deployment of security solutions and protection mechanisms provided by multiple third-party vendors, coping with the reduction of the efforts and resources invested in reinventing the wheel when developing an already available security application. Although SDN and NFV evolve as essential technologies for the development of next generation telecommunications, there have not been sufficient efforts in the development and introduction of standard binding interfaces between both SDN and NFV paradigms. In (Hyun et al., 2018), a first approach to the development of standard interfaces between SDN and cloud services shows how such interfaces collaborate with the controller in the orchestration of different security functionalities as instructed by security policies that monitor and control the network conditions. The evolution of telecommunications often requires the integration of elements and aspects from emerging tech-

nologies. Standardizing the binding interfaces to take advantage of the interaction between such technologies can greatly help in the improvement and promotion of inherent concepts like security in this particular case.

6.2. SDN security assessment

In programmable and flexible architectures like SDN it is common that operators are tempted to integrate vast and complex solutions to mitigate certain situations. However, they often disregard the effects and the changes in the network dynamics introduced by the integration of such solutions. When new security elements are included in the system, the individual assessment is in general focused on checking how the solution reacts to the situation it is intended for, without covering aspects related to the effects generated by both the deployment of the element itself and the integration to other members in the system. In particular, a situation that exhibits a behavior similar to the one detailed above occurs in the deployment of multiple security applications along with a policy-based scheme. It often happens that policies conflict with the security measures provided by the applications, causing the flow-entry replacement phenomenon in the switches and as a consequence leaving the switches vulnerable to the attacks that the replaced entries were supposed to mitigate. In a case like the former one, it is important to monitor the dynamic evolution of the network as it passes by different stages in its lifetime; regarding security there must be a mechanism that either matches the current network state to a detailed template that describes the ideal security state or that contains certain security artifacts that account the changes observed in certain timespan and enforces a set of tests and actions that guard the particular security conditions that should be maintained to keep the network secure.

6.3. SDN forensics

Determining the root cause and the source of attacks in SDN is a challenging task and the methods to gather attack evidence utilized in conventional networks are not suitable in the SDN context (Khan et al., 2016). Attack detection engines usually act over data collected on the network element that has been compromised or on the interface that is transferring the anomalous flows, detection and prevention engines hardly invest any effort on following the evidence trails left by the attack on preceding network elements straight back to the attack source itself. Unfortunately, a network application to track down all evidence along the attack flow path might need the enforcement of complex algorithms, attack features database, and perhaps machine learning instances which in practice reflects in a resources and time consuming task not to mention the added complexity to achieve a seamless network operation due to the pervasive characteristic of such approach. Forensics play an important role as key enabler of some other security mechanisms since all results from analysis and the evidence collected can be used as source for the development of new protection strategies, for instance, relevant attack data resulting from forensics processing might be leveraged for the design of and evolution of the set of security policies, or also used as input for training of machine learning approaches. The packet evidence gathered might be used as matching criteria for deep packet inspection mechanisms. Once identified the sources of attacks, the network can be reconfigured to stop such sources from causing harm to the network and to sanitize the interfaces and elements that were compromised during the attack.

6.4. Network resilience

A fault tolerant network approach might be capable of returning the network back to a stable operating state in the occurrence of an incident that leaves the network disabled or unable to work properly. Remarkable network incidents include: controller crash or shutdown; connectivity disruption; and energy outages (da Silva et al., 2015). SDN deployments are prone to suffer any of those incidents, particularly when the network is under certain specialized attack vector. As perceived in SDN security state-of-art, current research in SDN security focuses on predicting, preventing, protecting, detecting and reacting to security issues but very few proposals include mechanisms to recover the network state after a successful attack, only in (Sahay et al., 2017) path monitoring detects congested links in DoS/DDoS attack situations and a path computation scheme redistributes the congestion flows in alternative network paths. In spite of the single point of failure problem inherent to the SDN paradigm, insufficient works approach this problem or propose recovery schemes for controller outage. A complete security system might be proposed as a compound architecture that includes both proactive mechanisms that comprise the front line to act against the situations that compromise the security. And reactive mechanisms as well, to enforce actions that focus on recovering the network state once it has been compromised.

6.5. Trusted network applications

It was previously pointed out that network controllers can neither distinguish when network applications demand the installation of malicious flow rules in the infrastructure nor when certain applications interfere in the packet handling process exploiting vulnerabilities in the northbound APIs, preventing other applications from handling the packets they are intended to process. Besides, it was mentioned that although certain applications might not be designed to execute deviant behavior, in certain cases such applications innocently, when integrated to the system, release or enable vulnerabilities. From the perspective that the SDN is prone to both mentioned situations it is necessary to implement a security mechanism whose principal function should be to audit the security of network applications, in two phases. In one phase the application might be audited in isolation, as an individual component that will be later integrated to the SDN system, at this stage a vulnerability assessment and code inspection (coverage tests, unitary tests, anomalous code snippets, etc.) should be executed. In the other phase, a functional assessment might detect if the integration and interaction of the application with some other SDN instances induces any undesired behavior, for instance, if the application conflicts with network policies when instructing the installation of flow rules. Suggested measures should be complemented with a trust authentication and authorization mechanism, therefore restricting the execution and access to the network to ill-intentioned third-party applications or shellcodes injected by means of another vulnerable services.

6.6. Programmable data planes

Several works reviewed in this article have introduced the implementation of the functionality of security hardware appliances by leveraging the features present in SDN. However, the separation of control and data planes is impractical when such security services need to be deployed on large and delay sensitive networks like in datacenters and cloud environments due to certain shortcomings that can be attributed to the centralized architecture in SDN. In the one hand, all flows from all switches need to be forwarded to the controller, in large networks such volumes of traffic reaching the controller can become a performance bottleneck. On the other hand, several security appliances, like the stateful firewalls, require historical network state information that in current SDN lies entirely in the network controller and a significant communication overhead materialized when such information is shared with devices on the data plane (Caprolu et al., 2019, Chowdhary et al., 1811).

Programmable data planes promise to improve header matching flexibility and programmability as many decisions upon packet recep-

tion will be made at the switches leveraging custom packet-level programs (Pontarelli et al., 2017) that extract packet header bytes, perform computations, overwrite packet fields and store certain state information (Bifulco and Rétvári, 2018). As a consequence, switches will be able to make forwarding decisions without the need to establish communication with the controller (Caprolu et al., 2019). The network overhead can be significantly reduced because the data plane devices do not need to forward all traffic to the control plane and since many forwarding decisions can be taken directly in the switch then the packet forwarding time is reduced as there is no need to wait for instructions from the controller, and at the same time relieving the computational load at the controller (Hwang et al., 2018).

7. Conclusion

The SDN architecture is a revolution in network management and control, adding special features that enhance different network functions and at the same time provide with solutions to cumbersome issues present in conventional networks. The centralized control and network programmability in SDN cooperate in speeding-up the prototyping and development of network functions, in general, most of the network functions found in conventional architectures can be rendered in SDN in the form of simple software implementations. Network security is also in the scope of network innovation through the enforcement of SDN features; works and proposals in the state-of-the-art detail the increase in the enforcement of SDN features in the development and rendering of different network security functions.

Despite the introduction of new security schemes and the enhancement of the existing ones which in turn provides with new tools and mechanisms for stronger network security and protection, reliable security in SDN cannot be completely guaranteed. Moreover, the additional layers and interfaces in SDN seamlessly propitiate the emergence of new vulnerabilities and threats to security. This latter statement defines two research and development roadmaps in SDN security discipline, on one hand, a branch of research aimed at leveraging SDN constitutive features to enhance network security while on the other hand a branch focused on promoting a secure and dependable SDN architecture with its associated layers and interfaces.

Vulnerabilities and network attacks in SDN are increasingly complex and sophisticated. Therefore, enabling the emergence of new challenges that compel security research to shape the SDN architecture to: be in constant interaction with different technologies; and attempt to integrate elements and attributes that can be enforced in the construction of innovative and multidisciplinary security frameworks. Machine learning algorithms, Cloud services, virtualized network functions, are good examples of technologies that can be joint for the construction of the next generation SDN security environment.

Deployment of SDN in production networks is still a vision and a lot of work is still to be done in SDN security to become this vision into reality. There are essential security problems requiring attention and dormant issues yet to be uncovered, hence the research field is still wide open and every new contribution helps to close the gap between that vision and the reality. SDN security goal should aim in achieving an automated, self-constructive, self-auditable and self-diagnostic security framework wide enough to attend different situations and criticality levels, reconfigurable enough to effectively re-shape its structure according to the severity and the impact of the corresponding situation, and capable to both distinguish and resolve the inconsistencies residing in its own constitutive system structure that could trigger unsupervised and unpredictable anomalous performance.

Although almost a decade has passed since the first proposals in SDN security were published, there are several open challenges and topics that have not matured enough and extensive efforts are needed in those specific fields, e.g. SDN forensics, policy mediators, SDN debuggers and vulnerability discovery. Somehow the hype in SDN security research has led the scientific community into focusing in some trendy topics, e.g. machine learning-based detection engines, programmable data planes and virtualized security functions; leaving important security aspects in a slow evolution curve with very few works published and opening a gap in the development of new security strategies that not only attend such overrated issues but that are needed for the construction of robust and complete unified security frameworks.

Declaration of competing interest

None.

Acknowledgements

This paper has been partially supported by the project “Red temática CYTED 519RT0580” funded by the Ibero-American Science and Technology Program CYTED. This research was also partially supported by the Colombian Department of Science Technology and Innovation (Colciencias) and the Government of Antioquia, under the contract CT609 of 2019.

References

Abubakar, A., Pranggono, B., 2017. Machine learning based intrusion detection system for software defined networks. In: 2017 Seventh International Conference on Emerging Security Technologies. EST, pp. 138–143, https://doi.org/10.1109/EST.2017.8090413.

Ahmad, I., Namal, S., Ylianttila, M., Gurtov, A., 2015. Security in software defined networks: a survey. IEEE Commun. Surv. Tutor. 17 (4), 2317–2346.

Ajaeiya, G.A., Adalian, N., Elhajj, I.H., Kayssi, A., Chehab, A., 2017. Flow-based intrusion detection system for sdn. In: 2017 IEEE Symposium on Computers and Communications. ISCC, pp. 787–793, https://doi.org/10.1109/ISCC.2017.8024623.

Akhunzada, A., Ahmed, E., Gani, A., Khan, M.K., Imran, M., Guizani, S., 2015. Securing software defined networks: taxonomy, requirements, and open issues. IEEE Commun. Mag. 53 (4), 36–44, https://doi.org/10.1109/MCOM.2015.7081073.

Akhunzada, A., Gani, A., Anuar, N.B., Abdelaziz, A., Khan, M.K., Hayat, A., Khan, S.U., 2016. Secure and dependable software defined networks. J. Netw. Comput. Appl. 61, 199–221.

Al-Shaer, E., Al-Haj, S., 2010. Flowchecker: configuration analysis and verification of federated openflow infrastructures. In: Proceedings of the 3rd ACM Workshop on Assurable and Useable Security Configuration. ACM, pp. 37–44.

AlEroud, A., Alsmadi, I., 2017. Identifying cyber-attacks on software defined networks: an inference-based intrusion detection approach. J. Netw. Comput. Appl. 80, 152–164, https://doi.org/10.1016/j.jnca.2016.12.024.

Ali, S.T., Sivaraman, V., Radford, A., Jha, S., 2015. A survey of securing networks using software defined networking. IEEE Trans. Reliab. 64 (3), 1086–1097, https://doi.org/10.1109/TR.2015.2421391.

Alsmadi, I., Xu, D., 2015. Security of software defined networks: a survey. Comput. Secur. 53, 79–108.

Banikazemi, M., Olshefski, D., Shaikh, A., Tracey, J., Wang, G., 2013. Meridian: an sdn platform for cloud network services. IEEE Commun. Mag. 51 (2), 120–127, https://doi.org/10.1109/MCOM.2013.6461196.

Battula, L.R., 2014. Network security function virtualization(nsfv) towards cloud computing with nfv over openflow infrastructure: challenges and novel approaches. In: 2014 International Conference on Advances in Computing, Communications and Informatics. ICACCI, pp. 1622–1628, https://doi.org/10.1109/ICACCI.2014.6968453.

Benton, K., Camp, L.J., Small, C., 2013. Openflow vulnerability assessment. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 151–152.

Bernardo, D.V., Chua, B.B., 2015. Introduction and analysis of sdn and nfv security architecture (sn-seca). In: 2015 IEEE 29th International Conference on Advanced Information Networking and Applications, pp. 796–801, https://doi.org/10.1109/AINA.2015.270.

Bianco, A., Birke, R., Giraudo, L., Palacin, M., 2010. Openflow switching: data plane performance. In: 2010 IEEE International Conference on Communications, pp. 1–5, https://doi.org/10.1109/ICC.2010.5502016.

Bifulco, R., Rtvri, G., 2018. A survey on the programmable data plane: abstractions architectures and open problems. In: Proc. IEEE HPSR. IEEE.

Braun, W., Menth, M., 2014. Software-defined networking using openflow: protocols, applications and architectural design choices. Future Internet 6 (2), 302–336.

Canini, M., Venzano, D., Peresini, P., Kostic, D., Rexford, J., 2012. A nice way to test openflow applications. In: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation. NSDI).

Caprolu, M., Raponi, S., Di Pietro, R., 2019. Fortress: an efficient and distributed firewall for stateful data plane sdn, security and communication networks. https://doi.org/10.1155/2019/6874592.

Chandrasekaran, B., Benson, T., 2014. Tolerating sdn application failures with legosdn. In: Proceedings of the 13th ACM Workshop on Hot Topics in Networks. ACM, pp. 1–7.

Cheminod, M., Durante, L., Seno, L., Valenza, F., Valenzano, A., Zunino, C., 2017. Leveraging sdn to improve security in industrial networks. In: 2017 IEEE 13th

International Workshop on Factory Communication Systems. WFCS, pp. 1–7, https://doi.org/10.1109/WFCS.2017.7991960.
A. Chowdhary, D. Huang, A. Alshamrani, A. Sabur, M. H. Kang, A. Kim, A. Velazquez, Sdfw: Sdn-based stateful distributed firewall, CoRR abs/1811.00634. arXiv:1811.00634. URL http://arxiv.org/abs/1811.00634.
Chung, C.-J., Khatkar, P., Xing, T., Lee, J., Huang, D., 2013. Nice: network intrusion detection and countermeasure selection in virtual network systems. IEEE Trans. Dependable Secure Comput. 10 (4), 198–211.
Conti, M., De Gaspari, F., Mancini, L.V., 2017. Know your enemy: stealth configuration-information gathering in sdn. In: International Conference on Green, Pervasive, and Cloud Computing. Springer, pp. 386–401.
Conti, M., Gaspari, F.D., Mancini, L.V., 2018. A novel stealthy attack to gather sdn configuration-information. IEEE Trans. Emerg. Top. Comput. 1–12, https://doi.org/10.1109/TETC.2018.2806977.
Coughlin, M., 2014. A Survey of Sdn Security Research. University of Colorado Boulder.
Cox, J.H., Clark, R., Owen, H., 2017. Leveraging sdn and webrtc for rogue access point security. IEEE Trans. Netw. Serv. Manag. 14 (3), 756–770, https://doi.org/10.1109/TNSM.2017.2710623.
da Silva, A.S., Smith, P., Mauthe, A., Schaeffer-Filho, A., 2015. Resilience support in software-defined networking: a survey. Comput. Network. 92, 189–207, https://doi.org/10.1016/j.comnet.2015.09.012.
Dacier, M., Dietrich, S., Kargl, F., Knig, H., et al., 2016. Network attack detection and defense: security challenges and opportunities of software-defined networking. Dagstuhl Rep. 6 (9), 1–28.
Dacier, M.C., Knig, H., Cwalinski, R., Kargl, F., Dietrich, S., 2017. Security challenges and opportunities of software-defined networking. IEEE Secur. Priv. 15 (2), 96–100, https://doi.org/10.1109/MSP.2017.46.
Deng, J., Hu, H., Li, H., Pan, Z., Wang, K.C., Ahn, G.J., Bi, J., Park, Y., 2015. Vnguard: an nfv/sdn combination framework for provisioning and managing virtual firewalls. In: 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), pp. 107–114, https://doi.org/10.1109/NFV-SDN.2015.7387414.
Deng, S., Gao, X., Lu, Z., Gao, X., 2018. Packet injection attack and its defense in software-defined networks. IEEE Trans. Inf. Forensics Secur. 13 (3), 695–705, https://doi.org/10.1109/TIFS.2017.2765506.
Dhawan, M., Poddar, R., Mahajan, K., Mann, V., 2015. Sphinx: detecting security attacks in software-defined networks. In: Proceedings of the 2015 Network and Distributed System Security (NDSS) Symposium.
Dong, P., Du, X., Zhang, H., Xu, T., 2016. A detection method for a novel ddos attack against sdn controllers by vast new low-traffic flows. In: 2016 IEEE International Conference on Communications. ICC, pp. 1–6, https://doi.org/10.1109/ICC.2016.7510992.
Dover, J.M., 2013. A Denial of Service Attack against the Open Floodlight Sdn Controller.
Dover, J.M., 2014. A Switch Table Vulnerability in the Open Floodlight Sdn Controller.
Farhady, H., Lee, H., Nakao, A., 2015. Software-defined networking: a survey. Comput. Network. 81, 79–95.
Feamster, N., Rexford, J., Zegura, E., 2013. The road to sdn. Queue 11 (12), 20.
Fernandez, M.P., 2013. Comparing openflow controller paradigms scalability: reactive and proactive. In: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications. AINA), pp. 1009–1016, https://doi.org/10.1109/AINA.2013.113.
Fonseca, P., Bennesby, R., Mota, E., Passito, A., 2012. A replication component for resilient openflow-based networking. In: 2012 IEEE Network Operations and Management Symposium, pp. 933–939, https://doi.org/10.1109/NOMS.2012.6212011.
Foster, N., Harrison, R., Freedman, M.J., Monsanto, C., Rexford, J., Story, A., Walker, D., 2011. Frenetic: a network programming language. ACM Sigplan Not. 46 (9), 279–291.
Fundation, O.N., 2012. Software-defined networking: the new norm for networks. ONF White Paper 2, 2–6.
Gray, N., Zinner, T., Tran-Gia, P., 2017. Enhancing sdn security by device fingerprinting. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management. IM, pp. 879–880, https://doi.org/10.23919/INM.2017.7987393.
Haleplidis, E., Salim, J.H., Halpern, J.M., Hares, S., Pentikousis, K., Ogawa, K., Wang, W., Denazis, S., Koufopavlou, O., 2015. Network programmability with forces. IEEE Commun. Surv. Tutor. 17 (3), 1423–1440, https://doi.org/10.1109/COMST.2015.2439033.
Hinrichs, T., Mitchell, J., Gude, N., Shenker, S., Casado, M., 2008. Expressing and Enforcing Flow-Based Network Security Policies. Tech. rep. University of Chicago.
Hizver, J., 2015. Taxonomic modeling of security threats in software defined networking. In: Proceedings of BlackHat Conference 2015, pp. 1–16.
Hogg, S., 2014. Sdn Security Attack Vectors and Sdn Hardening: Securing Sdn Deployments Right from the Start.
Hong, S., Xu, L., Wang, H., Gu, G., 2015. Poisoning network visibility in software-defined networks: new attacks and countermeasures. In: Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS), vol. 15, pp. 8–11.
Hu, H., Han, W., Ahn, G.-J., Zhao, Z., 2014a. Flowguard: building robust firewalls for software-defined networks. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking. ACM, pp. 97–102.
Hu, F., Hao, Q., Bao, K., 2014b. A survey on software-defined network and openflow: from concept to implementation. IEEE Commun. Surv. Tutor. 16 (4), 2181–2206, https://doi.org/10.1109/COMST.2014.2326417.
Hu, Z., Wang, M., Yan, X., Yin, Y., Luo, Z., 2015. A comprehensive security architecture for sdn. In: 2015 18th International Conference on Intelligence in Next Generation Networks, pp. 30–37, https://doi.org/10.1109/ICIN.2015.7073803.
Hussein, A., Elhajj, I.H., Chehab, A., Kayssi, A., 2016. Sdn security plane: an architecture for resilient security services. In: 2016 IEEE International Conference on Cloud Engineering Workshop. IC2EW, pp. 54–59, https://doi.org/10.1109/IC2EW.2016.15.
Hwang, R.-H., Nguyen, V.-L., Lin, P.-C., 2018. Statefit: a security framework for sdn programmable data plane model. In: 2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN). IEEE, pp. 168–173.
Hyun, S., Kim, J., Kim, H., Jeong, J., Hares, S., Dunbar, L., Farrel, A., 2018. Interface to network security functions for cloud-based security services. IEEE Commun. Mag. 56 (1), 171–178, https://doi.org/10.1109/MCOM.2018.1700662.
Jain, R., Paul, S., 2013. Network virtualization and software defined networking for cloud computing: a survey. IEEE Commun. Mag. 51 (11), 24–31, https://doi.org/10.1109/MCOM.2013.6658648.
Jarraya, Y., Madi, T., Debbabi, M., 2014. A survey and a layered taxonomy of software-defined networking. IEEE Commun. Surv. Tutor. 16 (4), 1955–1980, https://doi.org/10.1109/COMST.2014.2320094.
Jarschel, M., Oechsner, S., Schlosser, D., Pries, R., Goll, S., Tran-Gia, P., 2011. Modeling and performance evaluation of an openflow architecture. In: Proceedings of the 23rd International Teletraffic Congress, International Teletraffic Congress, pp. 1–7.
Jo, H., Nam, J., Shin, S., 2018. Nosarmor: Building a Secure Network Operating System, Security and Communication Networks.
Jouini, M., Rabai, L.B.A., Aissa, A.B., 2014a. Classification of security threats in information systems. Procedia Comput. Sci. 32, 489–496.
Jouini, M., Rabai, L.B.A., Aissa, A.B., 2014b. Classification of security threats in information systems. Procedia Comput. Sci. 32, 489–496.
Kaur, S., Singh, J., Ghumman, N.S., 2014. Network programmability using pox controller. In: ICCCS International Conference on Communication, Computing & Systems, IEEE, vol. 138, .
Kendall, K.K.R., 1999. A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. Ph.D. thesis. Massachusetts Institute of Technology.
Khan, S., Gani, A., Wahab, A.W.A., Abdelaziz, A., Ko, K., Khan, M.K., Guizani, M., 2016. Software-defined network forensics: motivation, potential locations, requirements, and challenges. IEEE Network 30 (6), 6–13, https://doi.org/10.1109/MNET.2016.1600051NM.
Khurshid, A., Zhou, W., Caesar, M., Godfrey, P., 2012. Veriflow: verifying network-wide invariants in real time. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. ACM, pp. 49–54.
Kim, H., Feamster, N., 2013. Improving network management with software defined networking. IEEE Commun. Mag. 51 (2), 114–119, https://doi.org/10.1109/MCOM.2013.6461195.
Kreutz, D., Ramos, F., Verissimo, P., 2013. Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 55–60.
Kreutz, D., Ramos, F.M.V., Verssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S., 2015. Software-defined networking: a comprehensive survey. Proc. IEEE 103 (1), 14–76, https://doi.org/10.1109/JPROC.2014.2371999.
Kreutz, D., Verssimo, P.J.E., Magalhaes, C., Ramos, F.M.V., 2018. The kiss principle in software-defined networking: a framework for secure communications. IEEE Secur. Priv. 16 (5), 60–70, https://doi.org/10.1109/MSP.2018.3761717.
Lara, A., Kolasani, A., Ramamurthy, B., 2014a. Network innovation using openflow: a survey. IEEE Commun. Surv. Tutor. 16 (1), 493–512, https://doi.org/10.1109/SURV.2013.081313.00105.
Lara, A., Kolasani, A., Ramamurthy, B., 2014b. Network innovation using openflow: a survey. IEEE Commun. Surv. Tutor. 16 (1), 493–512, https://doi.org/10.1109/SURV.2013.081313.00105.
Le, A., Dinh, P., Le, H., Tran, N.C., 2015. Flexible network-based intrusion detection and prevention system on software-defined networks. In: 2015 International Conference on Advanced Computing and Applications. ACOMP, pp. 106–111, https://doi.org/10.1109/ACOMP.2015.19.
Lee, W., Kim, N., 2017. Security policy scheme for an efficient security architecture in software-defined networking. Information 8 (2), 65.
Lee, S., Kim, J., Shin, S., Porras, P., Yegneswaran, V., 2017a. Athena: a framework for scalable anomaly detection in software-defined networks. In: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. DSN, pp. 249–260, https://doi.org/10.1109/DSN.2017.42.
Lee, S., Yoon, C., Lee, C., Shin, S., Yegneswaran, V., Porras, P., 2017b. Delta: a security assessment framework for software-defined networks. In: Proceedings of the 2017 Network and Distributed System Security (NDSS) Symposium, vol. 17, .
Leng, J., Zhou, Y., Zhang, J., Tang, Y., Chen, K., 2018. Exploiting the Vulnerability of Flow Table Overflow in Software-Defined Network: Attack Model, Evaluation, and Defense, Security and Communication Networks. , https://doi.org/10.1155/2018/4760632.
Li, C.-S., Liao, W., 2013. Software defined networks. IEEE Commun. Mag. 51 (2) 113113.
Lin, Z., Tao, D., Wang, Z., 2017. Dynamic construction scheme for virtualization security service in software-defined networks. Sensors 17 (4), 920.
Lindqvist, U., Jonsson, E., 1997. How to systematically classify computer security intrusions. In: Proceedings. 1997 IEEE Symposium on Security and Privacy. IEEE, pp. 154–163. Cat. No. 97CB36097.
Liyanage, M., Ylianttila, M., Gurtov, A., 2014. Securing the control channel of software-defined mobile networks. In: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, pp. 1–6, https://doi.org/10.1109/WoWMoM.2014.6918981.
Loch, K.D., Carr, H.H., Warkentin, M.E., 1992. Threats to information systems: today's reality, yesterday's understanding. MIS Q. 16 (2), 173–186, https://doi.org/10.2307/249574.
Lorenz, C., Hock, D., Scherer, J., Durner, R., Kellerer, W., Gebert, S., Gray, N., Zinner, T.,

Tran-Gia, P., 2017. An sdn/nfv-enabled enterprise network architecture offering fine-grained security policy enforcement. IEEE Commun. Mag. 55 (3), 217–223, https://doi.org/10.1109/MCOM.2017.1600414CM.
Matias, J., Garay, J., Toledo, N., Unzilla, J., Jacob, E., 2015. Toward an sdn-enabled nfv architecture. IEEE Commun. Mag. 53 (4), 187–193, https://doi.org/10.1109/MCOM.2015.7081093.
Mattos, D.M.F., Duarte, O.C.M.B., 2016. Authflow: authentication and access control mechanism for software defined networking. Ann. Telecommun. 71 (1112), 607–615.
McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. Openflow: enabling innovation in campus networks. Comput. Commun. Rev. 38 (2), 69–74.
Nguyen, T.-H., Yoo, M., 2017. Analysis of link discovery service attacks in sdn controller. In: 2017 International Conference on Information Networking. ICOIN, pp. 259–261, https://doi.org/10.1109/ICOIN.2017.7899515.
Nunes, B.A.A., Mendonca, M., Nguyen, X.N., Obraczka, K., Turletti, T., 2014. A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. 16 (3), 1617–1634, https://doi.org/10.1109/SURV.2014.012214.00180.
Oehlert, P., 2005. Violating assumptions with fuzzing. IEEE Secur. Priv. 3 (2), 58–62, https://doi.org/10.1109/MSP.2005.55.
Omnes, N., Bouillon, M., Fromentoux, G., Grand, O.L., 2015. A programmable and virtualized network it infrastructure for the internet of things: how can nfv sdn help for facing the upcoming challenges. In: 2015 18th International Conference on Intelligence in Next Generation Networks, pp. 64–69, https://doi.org/10.1109/ICIN.2015.7073808.
Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J.J., Lorca, J., Folgueira, J., 2017. Network slicing for 5g with sdn/nfv: concepts, architectures, and challenges. IEEE Commun. Mag. 55 (5), 80–87, https://doi.org/10.1109/MCOM.2017.1600935.
Pfaff, B., Lantz, B., Heller, B., et al., 2014. Openflow Switch Specification. version 1.5. 0.
Pontarelli, S., Bonola, M., Bianchi, G., 2017. Smashing sdn built-in actions: programmable data plane packet manipulation in hardware. In: 2017 IEEE Conference on Network Softwarization (NetSoft). IEEE, pp. 1–9.
Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., Gu, G., 2012. A security enforcement kernel for openflow networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. ACM, pp. 121–126.
Porras, P.A., Cheung, S., Fong, M.W., Skinner, K., Yegneswaran, V., 2015. Securing the software defined network control layer. In: Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS). Internet Society, pp. 1–15.
Qiu, X., Zhang, K., Ren, Q., 2017. Global flow table: a convincing mechanism for security operations in sdn. Comput. Network. 120, 56–70, https://doi.org/10.1016/j.comnet.2017.04.002.
Ranjbar, A., Komu, M., Salmela, P., Aura, T., 2016. An sdn-based approach to enhance the end-to-end security: ssl/tls case study. In: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, pp. 281–288, https://doi.org/10.1109/NOMS.2016.7502823.
Rawat, D.B., Reddy, S.R., 2017. Software defined networking architecture, security and energy efficiency: a survey. IEEE Commun. Surv. Tutor. 19 (1), 325–346, https://doi.org/10.1109/COMST.2016.2618874.
Rpke, C., 2016. Sdn Malware: Problems of Current Protection Systems and Potential Countermeasures. Sicherheit, Sicherheit, Schutz und Zuverlssigkeit.
Rpke, C., Holz, T., 2015. Sdn rootkits: subverting network operating systems of software-defined networks. In: International Workshop on Recent Advances in Intrusion Detection. Springer, pp. 339–356.
Sahay, R., Blanc, G., Zhang, Z., Toumi, K., Debar, H., 2017. Adaptive policy-driven attack mitigation in sdn. In: Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures. ACM, p. 4.
Sama, M.R., Said, S.B.H., Guillouard, K., Suciu, L., 2014. Enabling network programmability in lte/epc architecture using openflow. In: 2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks. WiOpt, pp. 389–396, https://doi.org/10.1109/WIOPT.2014.6850324.
Schehlmann, L., Abt, S., Baier, H., 2014. Blessing or curse? revisiting security aspects of software-defined networking. In: 10th International Conference on Network and Service Management (CNSM) and Workshop, pp. 382–387, https://doi.org/10.1109/CNSM.2014.7014199.
Scott-Hayward, S., 2015. Design and deployment of secure, robust, and resilient sdn controllers. In: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), pp. 1–5, https://doi.org/10.1109/NETSOFT.2015.7258233.
Scott-Hayward, S., O'Callaghan, G., Sezer, S., 2013. Sdn security: a survey. In: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1–7, https://doi.org/10.1109/SDN4FNS.2013.6702553.
Scott-Hayward, S., Natarajan, S., Sezer, S., 2016. A survey of security in software defined networks. IEEE Commun. Surv. Tutor. 18 (1), 623–654.
Sezer, S., Scott-Hayward, S., Chouhan, P.K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., Rao, N., 2013. Are we ready for sdn? implementation challenges for software-defined networks. IEEE Commun. Mag. 51 (7), 36–43, https://doi.org/10.1109/MCOM.2013.6553676.
A. Shaghaghi, M. A. Kafar, R. Buyya, S. Jha, Software-defined network (SDN) data plane security: Issues, solutions and future directions, CoRR abs/1804.00262. arXiv:1804.00262. URL http://arxiv.org/abs/1804.00262.
Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2013a. Avant-guard: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security. ACM, pp. 413–424.
Shin, S., Porras, P.A., Yegneswaran, V., Fong, M.W., Gu, G., Tyson, M., 2013b. Fresco: modular composable security services for software-defined networks. In: Proceedings of the 2013 Network and Distributed System Security Symposium (NDSS). Internet Society.
Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., Yegneswaran, V., Noh, J., Kang, B.B., 2014. Rosemary: a robust, secure, and high-performance network operating system. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 78–89.
Shin, S., Xu, L., Hong, S., Gu, G., 2016. Enhancing network security through software defined networking (sdn). In: 2016 25th International Conference on Computer Communication and Networks. ICCCN, pp. 1–9, https://doi.org/10.1109/ICCCN.2016.7568520.
Shu, Z., Wan, J., Li, D., Lin, J., Vasilakos, A.V., Imran, M., 2016. Security in software-defined networking: threats and countermeasures. Mobile Network. Appl. 21 (5), 764–776.
Simmons, C., Ellis, C., Shiva, S., Dasgupta, D., Wu, Q., 2014. Avoidit: a cyber attack taxonomy. In: 9th Annual Symposium on Information Assurance. ASIA14, pp. 2–12.
Skowyra, R.W., Lapets, A., Bestavros, A., Kfoury, A., 2013. Verifiably-safe software-defined networks for cps. In: Proceedings of the 2nd ACM International Conference on High Confidence Networked Systems. ACM, pp. 101–110.
Sloan, R.H., Warner, R., 2013. Unauthorized Access: the Crisis in Online Privacy and Security. CRC press.
Stallings, W., 2013. Software-defined networks and openflow. Inter. Protocol J. 16 (1), 2–14.
Tantar, E., Tantar, A.-A., Kantor, M., Engel, T., 2018. On using cognition for anomaly detection in sdn. In: EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI. Springer, pp. 67–81.
Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., Sherwood, R., 2012. On controller performance in software-defined networks., Hot-ICE, vol. 12, pp. 1–6.
Toseef, U., Zaalouk, A., Rothe, T., Broadbent, M., Pentikousis, K., 2014. C-bas: certificate-based aaa for sdn experimental facilities. In: 2014 Third European Workshop on Software Defined Networks (EWSDN). IEEE, pp. 91–96.
Van der Merwe, J., Kalmanek, C., 2007. Network programmability is the answer. In: Workshop on Programmable Routers for the Extensible Services of Tomorrow (PRESTO 2007), Princeton, NJ.
Vaughan-Nichols, S.J., 2011. Openflow: the next generation of the network? Computer 44 (8), 13–15.
Wen, X., Chen, Y., Hu, C., Shi, C., Wang, Y., 2013. Towards a secure controller platform for openflow applications. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. ACM, pp. 171–172.
Wrona, K., Oudkerk, S., Szwaczyk, S., Amanowicz, M., 2016. Content-based security and protected core networking with software-defined networks. IEEE Commun. Mag. 54 (10), 138–144, https://doi.org/10.1109/MCOM.2016.7588283.
Wrona, K., Amanowicz, M., Szwaczyk, S., Gierowski, K., 2017. Sdn testbed for validation of cross-layer data-centric security policies. In: 2017 International Conference on Military Communications and Information Systems. ICMCIS, pp. 1–6, https://doi.org/10.1109/ICMCIS.2017.7956483.
Xing, T., Huang, D., Xu, L., Chung, C.J., Khatkar, P., 2013. Snortflow: a openflow-based intrusion prevention system in cloud environment. In: 2013 Second GENI Research and Educational Experiment Workshop, pp. 89–92, https://doi.org/10.1109/GREE.2013.25.
Yan, Z., Zhang, P., Vasilakos, A.V., 2016. A security and trust framework for virtualized networks and software-defined networking. Secur. Commun. Network. 9 (16), 3059–3069.
Yoon, C., Lee, S., Kang, H., Park, T., Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2017a. Flow wars: systemizing the attack surface and defenses in software-defined networks. IEEE/ACM Trans. Netw. 25 (6), 3514–3530, https://doi.org/10.1109/TNET.2017.2748159.
Yoon, C., Shin, S., Porras, P., Yegneswaran, V., Kang, H., Fong, M., O'Connor, B., Vachuska, T., 2017b. A security-mode for carrier-grade sdn controllers. In: Proceedings of the 33rd Annual Computer Security Applications Conference. ACM, pp. 461–473.
Zhang, S.-h., Meng, X.-x., Wang, L.-h., 2017. Sdnforensics: a comprehensive forensics framework for software defined network. Development 3 (4), 5.

Juan Camilo Correa Chica, received the Bs. Eng in Electronics Engineering in 2010 from the University of Antioquia and received his MSc degree in Telecommunications Engineering from the same university in 2016. He is a lecturer and researcher at Instituto Tecnológico Metropolitano (ITM) and has been working as part time lecturer for digital systems design and software programming courses for the University of Antioquia. His research interests include: simulation and modeling of telecommunication systems; Internet of Things; Big Data; Software Defined Networking; and issues regarding software engineering such as efficient algorithms, data structures, optimization and metaheuristics. ORCID: 0000-0003-3476-9312

Jenny Cuatindioy Imbachi, is a Professor at the Telecommunications Engineering Department at the University of Medellín, Medellín, Colombia. She received her Bs. Eng in Electronics and Telecommunications engineering back in 1998 from Universidad del Cauca. She pursued a Specialization in Networks and Telecommunications Services at Universidad del Cauca in 2005. She also received her M.Sc. degree in Telecommunications engineering from Universidad de Antioquia in 2017. Her main research interests include Channel Coding, Optical Networks, Access and Carrier Grade Networks and Next-Gen Networks.

Juan Felipe Botero Vega, is a Professor at the Electronics and Telecommunications Engineering Department at the University of Antioquia, Medellín, Colombia. In 2006 he received his Computer Science Degree from the University of Antioquia, his M.Sc. degree in Telematics Engineering in 2008 from the Technical University of Catalonia, UPC, in Barcelona, Spain, and his Ph.D. degree in Telematics Engineering at UPC. In 2013, he joined the research group on applied telecommunications (GITA) at the Electronics and Telecommunications Engineering Department. His main research interests include Quality of Service, Software Defined Networking, Network Virtualization, Data Center Network Virtualization and resource allocation in virtual networks. (https://sites.google.com/site/juanfebotero/). ORCID: 0000-0002-7072-8924.