Received August 7, 2021, accepted August 26, 2021, date of publication September 1, 2021, date of current version September 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3109564

A Survey of the Main Security Issues and Solutions for the SDN Architecture

MARÍA B. JIMÉNEZ[1], DAVID FERNÁNDEZ[1], JORGE EDUARDO RIVADENEIRA[2], (Member, IEEE), LUIS BELLIDO[1], AND ANDRÉS CÁRDENAS[1], (Member, IEEE)
[1]Department of Telematic Systems Engineering, ETSI Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain
[2]Centre for Informatics and Systems of the University of Coimbra, 3000-214 Coimbra, Portugal

Corresponding author: María B. Jiménez (mb.jimenez@alumnos.upm.es)

This work was supported in part by the Ecuadorian Funding Institute SENESCYT-Secretaría de Educación Superior, Ciencia, Tecnología e Innovación, under the Ph.D. Grant CZ02-000486-2019, in part by the Spanish Ministry of Science and Innovation in the context of the ECTICS Project under Grant PID2019-105257RB-C21, and in part by the Spanish Ministry of Science, Innovation and Universities in the context of the Go2Edge Project under Grant RED2018-102585-T.

ABSTRACT The software-defined networking (SDN) paradigm proposes the decoupling of control and data planes and a centralized software-oriented management approach based on a central controller, easing the development of new applications and services. These design principles pave the way for a more flexible, fast, and dynamic software-controlled network. However, in terms of security, the elements that comprise the SDN architecture present several vulnerabilities, which could be exploited by attackers to carry out malicious actions and thus affect the network and its services. Although for several years, some studies have already focused on identifying the weaknesses of the SDN layer structure, the nature of the attacks, and possible solutions for this paradigm, the literature contains few contributions that review and discuss this topic in an integral fashion. This paper provides a comprehensive, updated, and detailed review of the main security issues and mitigating measures for all layers and interfaces of the SDN architecture, classifying the contributions according to the STRIDE threat modeling methodology categories. Finally, this manuscript identifies, discusses, and synthesizes open challenges and future research directions in this area.

INDEX TERMS SDN interfaces, SDN planes, SDN security, STRIDE.

I. INTRODUCTION

The software-defined networking (SDN) paradigm is deeply transforming telecommunications networks and has been broadly adopted as an enabling technology in initiatives, such as 5G or the Internet of Everything (IoE) [1], [2]. Although it has been more than two decades since its inception, SDN is constantly evolving, and there are an increasing number of requirements from the technology world, demanding that SDN networks be more dynamic, flexible, and secure.

From 2020 to 2025, the use of SDN networks is expected to increase by 19% [3]. There are several boosters influencing this growth, but much of it is undoubtedly driven by cloud service providers (CSPs), who have seen in the innovation that SDN, in relation to traditional networks, represents a solution to build highly scalable, reliable, and automated data-center infrastructures [3]. The SDN architecture is based on the decoupling of the data plane from the control plane, with the controller as the main actor. The controller is the most sensitive part of the whole architecture and interacts with the data plane through the southbound interface and with applications through the northbound interface. Additionally, the east-/westbound interface is responsible for interconnecting distributed controllers [4], [5].

This architecture offers several advantages in terms of infrastructure management and growth projection. Centralized control provides a broad and detailed view of the network and facilitates the provision of services. Simultaneously, the open architecture together with the tendency to use open-source software offers a wide range of possibilities concerning applications, allowing their development by third parties, which translates into an economic benefit.

In contrast to the mentioned advantages, the SDN architecture presents some vulnerabilities [15], [16]. In fact, the Open Networking Foundation (ONF) issued a paper in which the main security challenges faced by SDN architecture are

The associate editor coordinating the review of this manuscript and approving it for publication was Mehdi Sookhak.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

TABLE 1. Related work comparison.

References	App Plane	NBI	Control Plane	EWBI	Stateless DP	Stateful DP	SBI	Security solutions' detail level
[6]	✓		✓		✓			H
[7]	✓	✓	✓	✓	✓		✓	L
[8]			✓					L
[9]			✓		✓			H
[10]						✓		L
[11]			✓					M
[12]	✓		✓		✓	✓	✓	M
[13]	✓		✓		✓			L
[14]			✓					M
This survey	✓	✓	✓	✓	✓	✓	✓	**H**

L=low, M=medium, H=high

exposed and in turn, the organization proposed a set of principles in that area [17]. This ONF manuscript emphasizes the following: the SDN security challenges that may be present when there is centralized control; the dual effects of having programmability in the network; the complications of integrating legacy protocols, such as DNS, NAT, or BGP, into the SDN architecture without a prior compatibility analysis of security aspects; and finally, the lack of trust relationship that may exist in the interconnection between different domains. Additionally, there is great concern that globally, the number of cybersecurity breaches has increased by 11% since the beginning of 2018 and is projected to grow in the coming years [18], which can significantly affect SDN network performance.

Adversaries can exploit these security gaps by launching different types of attacks. The origin of these attacks does not necessarily come from an external source, such as a distributed denial-of-service attack (DDoS) executed by means of a botnet. There are also attacks that can take advantage of vulnerabilities inherent in the layered architecture. For instance, among the vulnerable points are the communication channels, through which an attacker can access the controller and perform malicious actions, putting network functionality at risk.

Given this, both academia and industry are constantly working in the SDN security world; therefore, it is necessary to present the recent technological advances used to address security issues related to SDN architecture (e.g., mechanisms, techniques, or tools) in a concrete and unified way, in the expectation that further research will improve the ability of response to open challenges.

A. RELATED LITERATURE AND CONTRIBUTIONS

In recent years, work has been done on security mechanisms and solutions for the availability, confidentiality, and integrity of the elements that compose the SDN architecture, with reviews conducted by some authors. For example, in [6], several security aspects covering problems and solutions of SDN architecture layers are discussed. In [7], the authors conduct a broad and general review of SDN, including architecture security. However, the manuscript does not detail in depth the mechanisms used to solve the mentioned shortcomings. In [8], the authors expose the advances in SDN networks and their various applications, recognizing that SDN has been developed without taking into account several fundamental aspects of security, including both architecture security and measures to prevent and detect attacks. In this regard, the authors focus on the security of controllers. In [9], the process of network topology discovery in SDN and potential security issues are addressed. In [10], the authors try to raise awareness of the vulnerabilities that may exist in stateful data planes. In [11], some security threats to SDN controllers and mitigation techniques are considered. In [12], several security problems of the SDN architecture are presented, with a brief review of some solutions. In [13], the authors focus on the security and privacy aspects of 5G technology, exposing in a general way some problems of the SDN architecture, without detailing possible solutions. In [14], controller architectures are reviewed, taking into account several factors, one of which is security. However, to the best of our knowledge, no work has reviewed the most relevant security issues of all interfaces and layers of the SDN architecture (including the differentiation between stateless and stateful data planes) in a single manuscript or classifies in detail the solutions to these issues. Table 1 compares the most relevant works addressing security in SDN architecture in recent years with respect to this survey. For the abovementioned reasons, the objectives of this article are as follows:

- To review the main security issues of SDN architecture, along with proposed solutions for their detection or mitigation.
- To discuss the existing security aspects and the mechanisms employed in the solutions to the problems, identifying open issues that can be used as starting points for future research initiatives related to SDN security.

To perform this review, we considered all the interfaces and layers that constitute the SDN architecture, segmenting it into three blocks: a) the northbound interface and application plane, b) the east-/westbound interface and control plane, and c) the southbound interface and data plane. At the culmination of the review for each block, a summary table that includes the categories of the STRIDE threat modeling methodology will be presented [19]. For this document, the categories established in this methodology have been selected given their maturity in and adoption for evaluating security aspects of a system/infrastructure. In addition, the categories of this methodology are widely recognized, as it covers concepts from the cybersecurity world, such as spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privilege.

For the elaboration of this survey, articles were selected with the support of search tools, which allowed us to extract information from databases, such as IEEE Xplore, Scopus, Springer, and the ACM Digital Library. Although the searches with the tools mentioned yielded exorbitant amounts of papers relating to security solutions in the SDN world, it was important to distinguish between the following: a) security solutions for the SDN architecture, b) security solutions that use the SDN architecture or some of its components, and c) authors who mention SDN in their titles, although they use only some elements of the architecture to deploy tests, since their solutions are oriented to traditional networks.

Considering these particularities it is necessary to emphasise that this paper will solely address security solutions for the SDN architecture. In this sense, it was necessary to perform manual filtering by using keywords according to the interests of this document, until reaching the number of articles referenced in this work. For this purpose, the review of the abstract, research opportunities, and conclusions were used to determine whether or not to use the related articles.

B. PAPER ORGANIZATION

The rest of the paper is structured as follows. Section II defines the foundations of SDN, the main problems, and the recent security solutions presented. Section III will focus on the discussion and will address the open challenges to make these solutions available to both academia and industry. Finally, section IV will detail the conclusions of this paper.

II. SECURITY IN SDN ARCHITECTURE

BACKGROUND

Traditional networks have been a significant contribution to the telecommunications world. Nevertheless, due to the high demand for cloud services, traditional networks are becoming decreasingly flexible, programmable, and centrally managed for companies wishing to expand, such as telecommunications operators (TOs).

With this background, the SDN paradigm has been proposed, which aims to provide alternative solutions to the current limitations of traditional networks. The SDN concept originated approximately two decades ago. Thus, between 1990 and 2000, active networks appeared, which provided programmable functions. Subsequently, between 2001 and 2007, the separation of the data and control planes was observed through open interfaces; since 2007, work has been done on an open protocol to achieve communication between the mentioned layers [20]. Currently, the Open Networking Foundation (ONF), a nonprofit organization, is leading the SDN paradigm project. ONF has approximately

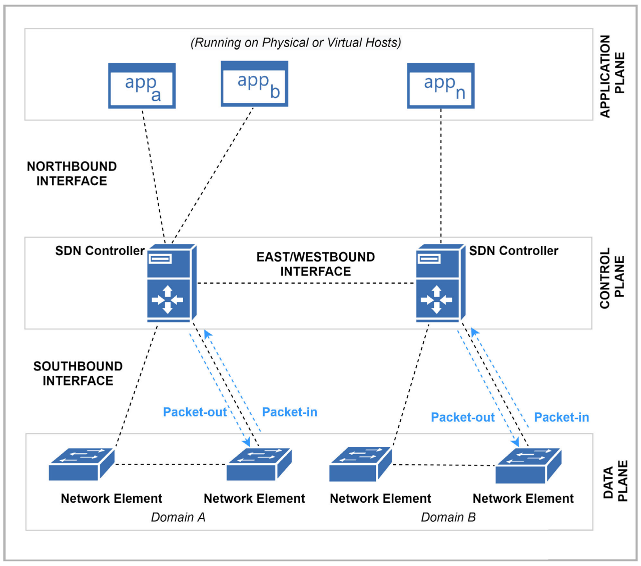

FIGURE 1. **SDN architecture. Adapted from [15].**

130 members, including companies, such as Facebook, Google, Deutsche Telekom, Microsoft, Yahoo!, and equipment manufacturers, among others [21].

According to the ONF, SDN presents an architecture composed of three planes or layers: the data or forwarding plane, the control plane, and the application plane [15]. The updating and exchange of information between the planes (data, control, application) are via interfaces. There are three interfaces in the SDN architecture: the southbound interface, which links the control plane to the data plane; the northbound interface, which links the control plane to the application plane; and the east-/westbound interface, which interconnects the distributed controllers. Figure 1 shows the SDN architecture explained above.

SDN provides a centralized view of the entire network through the controller, which simplifies the management of the nodes that compose it. It also makes it possible to maintain a considerable number of applications or network abstractions that can be developed by controller vendors or by third parties and that can be shared dynamically. All this can be reflected in reduced operating costs (OPEX) and capital expenditure (CAPEX) since it is no longer necessary to have a hardware or software manufacturer specialized in one brand.

Due to its advantages, the SDN architecture is currently being used in the deployment of faster services, for example, 5G networks [1]. However, it is recognized that in a large-scale implementation, it is necessary to define limits and security policies in the sharing of information and resources [15] because SDN, like many other technological paradigms, presents security threats, several of which are caused by its layered architecture [16].

A. NORTHBOUND INTERFACE AND APPLICATION PLANE

The application plane provides a set of programs (applications/abstractions) that are essential to satisfy the system's own needs, making it possible to generate or to respond to the requirements of the SDN environment through the controller. Currently, there is a wide spectrum of applications, including firewalls, routing policies, protocols, etc., which can be provided by the controller's developer itself or by third parties.

The communication between the application plane and the controller occurs via the northbound interface, which is not standardized; therefore, each controller defines its own primitive APIs and the kind to be used, e.g., RESTful API, programming languages, and specialized API (ad hoc) [22].

Even though, REST is currently the most commonly used API for applications in the industry, the ONF does not rule out the possibility of standardization; therefore, this organization is working on an open source API, which considers the participation of all stakeholders (application developers, controller providers, researchers, etc.) and in turn contributes in a better way to the generation of applications [23].

Since SDN applications are so diverse and are frequently implemented by third parties, they can pose security threats, since a malicious application could impersonate a legitimate application, enter the Controller and take actions on the configuration, even inserting false rules that completely modify the behavior of the network [24]. Specifically, concerning the variety of existing applications, security analyses have been carried out that identify several vulnerabilities that can lead to attacks [25]–[29]. However, to prevent this from happening, there are some solutions, for example Indago [30], which proactively detects malicious applications by using security-sensitive behavior graphs (SSBGs) and machine learning. According to the authors, their solution offers countermeasures for these attack vectors that involve information manipulation, impersonation, assignment of permissions, and information disclosure and can lead to a shutdown of the network service. Similarly, Shield [31] is a framework that mainly analyses the behavior of applications by employing the control-flow graph (CFG). With this solution, it is possible to identify malicious behavior that could lead to a modification of internal network parameters.

Something to keep in mind is that having multiple applications coming from third parties can cause interferences between them, generating conflicts in network policies [32]. Such interference can come from an attack or lead to one. Given these scenarios, there are proposals, such as MSAID [33], which through algorithms used on the same SDN application code, is capable of detecting interference; this is also the case for SAIDE [34], a proposal to detect and eliminate interference by using mathematical models.

With these premises, it can be argued that in the absence of the proper authentication and access control mechanisms that validate the origin of the applications, a northbound interface allows trust relationship attacks to be carried out towards the controller, many of which are derived from impersonation. At the same time, the absence of effective prior authentication and access control mechanisms for SDN applications exposes the network to handle illegitimate

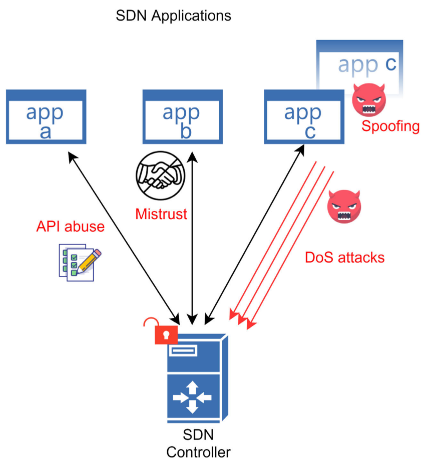

FIGURE 2. Associated attacks emerging from applications.

requests, generating increased consumption of resources and therefore resulting in network exhaustion. Precisely in [35], load tests, stress tests and DoS/DDoS attacks were carried out from the northbound interface, which led to a decrease in the performance of the analyzed controllers: POX [36], Ryu [37], Floodlight [38], ODL [39] and ONOS [40].

To solve authorization problems in applications, fine-grained and coarse-grained access controls have been proposed. Coarse-grained access controls are generally used to cover the perimeter vulnerabilities of a standalone system or application; in other words, they can be very useful in invariant environments where habitual behavior is almost predictive. Within this category, RBAC, MAC, and DAC, among others, can be mentioned. On the other hand, there are fine-grained access controls, that is, controls that have greater granularity and detail in application permissions, which are very useful in more dynamic environments.

Generally, the SDN architecture is used in multi-tenant, multiservice, multiprovider, multidomain environments; therefore, incorporating coarse-grained or very rigid access controls could lead to abuses in the use of permissions or could slow down network functionality. In fact, in [41], the authors show that one of the aforementioned coarse-grained access control mechanisms is not sufficient to control integrity attacks on the information flow, since it allows an attack called cross-app poisoning, in which an application accesses the controller and tricks other applications to execute actions.

In this sense, solutions for SDN application authorization problems are oriented towards fine-grained access control. Regarding authentication, there are solutions based on existing controllers or with independent mechanisms. With this background, the authors have made the following contributions.

In [42], the authors propose MD-UCON, which is an access control mechanism for the northbound interface, focusing on multidomain scenarios. This proposal uses RBAC mechanisms adapted for dynamic environments, applying a cross-domain role mapping method to support cross-domain access control. This solution is based on a model named UCON [43].

Through the use of blockchain, BlockAS [44] has been proposed to relate controllers with OpenFlow applications, using identifiers in the authentication and privilege assignment process. This proposal, in addition to providing AAA, offers decentralization and immutability of the database where application information, permissions and tokens are stored; it even allows the control of logs, in which the activity of all participants is recorded, thus enabling monitoring. In [45], a controller-independent solution is proposed to avoid the abuse of static permissions assigned to an API. It is composed of a northbound security extension, whose function is to repackage the built-in services of the northbound APIs, serving as a "mediator" between the OpenFlow applications and the controller. This solution also includes controller-specific IDS, which stores information about the permissions and accounting records of the OpenFlow application, and a high-level policy engine predefines the policies for each OpenFlow application. In this way, OpenFlow applications can be authenticated and authorized, and the legitimacy of accounting requests can be verified using password-based authentication and token-based authentication.

In [46], the authors propose the AEGIS framework, in which dynamic access control is maintained by verifying and monitoring API usage in real time. The components of this proposal include a data generator, which identifies the list of APIs to be protected; for this purpose, data extraction actions are performed using Daikon.[1] Similarly, there is a security rule generator, which defines the access rules between applications, APIs and their inputs or outputs. Finally, this framework includes a decision engine, which uses API hooking to intercept the behavior of the applications. Here, the input/output of each API is reviewed and contrasted with the rules.

BEAM [47] is a solution that assigns permissions to third-party applications. It is based on the network behavior, which is taken from metrics, such as flow_injection_rate or packet_in_rate, which are analyzed by an IDS, after which permissions to the applications are upgraded or downgraded in run-time. BEAM works with the following modules: registration handler, which is responsible for the registration and assignment of initial permissions to new applications; policy engine, which defines policies for upgrading/downgrading application permissions and has two important databases, namely, policy store and mapping table; and an activity detection module and an activity engine, which are modules responsible for reviewing application activity leveraged on an IDS and logs, respectively.

Tseng *et al.* [48], troubled with malicious application injection, DoS, and API abuse, proposed an architecture called SENAD, which is composed of four parts. The controller agents allow the interaction between the application plane controller (APC) and the data plane controller (DPC), based on a publish/subscribe model. The policy engine provides resource control for each application, as well as access control for them. The application sandbox and resource controller are responsible for isolating applications and delivering resources as required by the policy engine. Finally, the authentication and authorization modules work with the information exchanged between the APC and the DPC, handling password-based authentication and rules consulted with the policy engine for authorization.

In [49], an approach named the application authentication system and deployed outside the controller is proposed, allowing authentication, log history of unauthorized operations, and the management of access permissions, resources, application certificates, authorization elements, authentication and encryption, among others. Similarly, in [50], a security-as-a-service (SEaaS) solution is defined. One of its features is the authorization mechanism between the controller and the applications, taking the Floodlight controller architecture as a reference.

A web-based northbound interface framework influenced by REST is proposed in [51]. An API called "REST-like" is implemented since it is partially stateless[2]; that is, at one point, it stores information that it uses for the registration, authentication, and authorization of applications. It incorporates trust management, access control, and encrypted communication with the use of TLS certificates. The authors of [52] build on the above proposal to explore more "REST-like" functions that can be leveraged to provide security for OpenFlow applications.

Finally, in [53], the authors generate an architecture that only provides data to trusted third-party applications. For this purpose, the architecture makes use of the northbound interface with the NSS digital signature implementation and the NTRU encryption algorithm. Similarly, Hu *et al.* [54] propose a framework that contains two main modules: the permissions detection engine, which identifies the legality of applications' permissions, and the registration authorization engine, which performs both the registration and authorization of the application with the NTRU algorithm to avoid eavesdropping or tampering attacks.

Table 2 summarizes the proposed solutions to the main security problems for the northbound interface and the application plane. The table shows the reference of the paper, the issues that stimulated the research, the main mechanisms used in the solution approach and finally, a tabulation framed by the STRIDE categories; note that on many occasions, the authors provide solutions to more than one problem.

B. EAST/WESTBOUND INTERFACE AND CONTROL PLANE

SDN control is logically centralized in this plane through a "black box" named the controller or the network operating system, which manages requests made from the data plane.

[1]http://swmath.org/software/4319

[2]https://restfulapi.net/statelessness/

TABLE 2. Summary of contributions for the northbound interface and application plane.

Ref.	Addressed Problems	Main Solution Approaches	Literature Contribution					
			S	T	R	I	D	E
[30]	Malicious apps	SSBG + Machine learning	✓	✓		✓	✓	✓
[31]	Malicious apps	CFG behavior analysis		✓				
[42]	Lack of access control Cross-domain in multidomain scenarios	Modified RBAC Cross-domain role mapping method						✓
[44]	Illegal modification of essential database information Lack of assigning permissions control	Blockchain	✓	✓	✓	✓	✓	✓
[45]	API abuse	Northbound security extension IDS Policy engine	✓		✓	✓		✓
[46]	Lack of dynamic access control	Data Extraction: Daikon Decision engine: API hooking						✓
[47]	Lack of initial access permissions assignment to apps	Behavior-based access control method + IDS			✓			✓
[48]	API abuse, DoS, Malicious apps	Publish/Subscribe model	✓	✓			✓	✓
[49]	Lack of authentication mechanisms between the app and the Controller	Independent mechanism	✓		✓			✓
[50]	Lack of authentication mechanisms between the app and the Controller	Floodlight-based mechanism	✓					
[51] [52]	Nonencrypted communication between app and controller Lack of management over reliability Lack of access control	Use of REST-like functions	✓	✓	✓	✓		✓
[53]	Lack of third-party app authentication	NTRU encryption algorithm and NSS digital signatures	✓	✓				
[54]	Malicious apps	NTRU encryption algorithm	✓	✓				

S=spoofing, T=tampering, R=repudiation, I= information disclosure, D=denial of service, E=elevation of privilege

The controller has topology information, resource generation for applications, statistics, inventory, etc.

Currently, there are more than thirty controllers available, which have their own programming languages and interfaces; many are open source, whereas others are proprietary. Controllers can be classified as centralized architecture or distributed architecture [7], [55], [56]. According to the literature, there is a subclassification of distributed architectures, which can be flat or hierarchical. This subclassification is based on the responsibilities of each controller within the architecture [56]. Thus, in flat distributed architectures, all controllers have the same applications and responsibilities, whereas in hierarchical distributed architectures, there is a robust controller (root) that handles all applications and several controllers with fewer applications and therefore fewer responsibilities [55]. However, since the mode of communication between the planes, in both flat and hierarchical distributed architectures, is the same, for security issues, this differentiation is of little relevance; therefore, in this paper, the controllers will be referred to generically as centralized or distributed.

Centralized architectures use a single controller for the entire network to ease its management. Generally, this kind of architecture is used in implementations in which the throughput demand is low. However, relying on a unique control entity could create a single point of failure within the network. For instance, in peak traffic events, having a unique controller could create bottlenecks for incoming requests, affecting the time response. Moreover, from a security perspective, this architecture might be susceptible to DoS attacks [57].

On the other hand, the distributed architectures use multiple controllers in multidomain or heterogeneous networks [58]. Distributed controllers are used in large-scale deployments mainly by telecommunications operators for different functions, for example, for use in wide area networks (WANs).

Distributed controllers seek to improve conditions related to network scalability, throughput, latency, and resilience. Resilience is very effective in the presence of DoS attacks. This feature is handled by the controllers through fault tolerance mechanisms. In this way, Hyperflow [59], which uses the partition tolerance property of the WheelFS system, can be cited. In the failover and replication process, Ravana [60] and Pane [61] use ZooKeeper [62]. ONOS and ODL use the Raft consensus algorithm for the election of a leading controller [63]. In addition, [64] introduced a proposal in which the authors use anti-entropy methods through the Gossip protocol, achieving the detection of controllers with malicious traffic overload to subsequently choose a robust controller to lead the clustering against DDoS attacks in SDN networks.

Communication between controllers in distributed architectures is achieved through the east-/westbound interface, which, like the northbound interface, is not standardized; therefore, each controller proposes an interface. Interfaces oriented to the east provide communication between SDN controllers in different administrative domains of distributed environments, whereas interfaces oriented to the west interconnect traditional networks with SDN architectures [65]; although in [7], the authors refer that the eastbound interface is in charge of connecting SDN networks to traditional networks and the westbound interface is responsible for interconnecting SDN networks among them.

One of the principal security drawbacks in the east-/westbound interface is the lack of provenance verification of the information shared between controllers during the topology discovery update process. This issue might be exploited by attackers to hinder normal network performance [9]. In this context, the Internet Engineering Task Force (IETF) worked on SDNi [66], a protocol for the communication of the east/westbound interface. Although the SDNi protocol is currently in an expired state and presents a vulnerability that allows SQL injection [67], it is still relevant for the scientific community; in fact, there are proposals for interconnection in distributed environments, which refer to SDNi [68]–[71]. In the aforementioned references, no security factors have been specified, unlike [72], which proposes a multicontroller solution, making use of a layer responsible for concentrating access control and routing decisions, or [73], in which a distributed firewall service (DFS), a distributed load balancer service (DLBS), and channel assurance through SSL are implemented. On the other hand, for securing communication at the east-/westbound interface, there are solutions that employ identity-based cryptography (IBC) [74] or that provide a protected channel using elliptic curve cryptography (ECC) [75].

Even though some mechanisms contribute to the recovery of failures in SDN controllers, attacks do not cease and are presented each time in a different way. For this reason, there are entities dedicated to vulnerability and security analysis and that try to guide industry and academia towards the search for new and better security alternatives (e.g., Red Hat, OWASP,[3] academia). In this regard, in [76], the authors perform a security analysis of the following drivers: ODL, ONOS, Rosemary, and Ryu, looking at the security measures that each driver has. After this analysis, the authors determine that although there is no completely secure controller, the controllers with the best countermeasures against attacks are ODL followed by ONOS. In this sense, and due to the considerable presence of ONOS and ODL controllers in the literature, the following paragraphs will focus on them. Thus, the main security aspects of ONOS and ODL are presented below.

- The ODL controller is able to provide AAA service and secure network bootstrapping infrastructure modules to avoid authentication, repudiation, or impersonation problems, whether from users or applications. To avoid impersonation in host tracking, ODL uses parameters, such as MAC address, IP and location address, and VLAN ID, within the device manager instead of just using the MAC address. Through the unified secure channel feature it provides secure communication with TLS/DTLS support [77].
- The ONOS controller, is provided with a security mode-ONOS, which has fine-grained access control mechanisms for both applications and users, protecting the controller from an elevation of privilege attacks [78]. It also has a security audit service that prevents repudiation. It does not refer to a mechanism that prevents impersonation attacks in the host tracking process.

New studies identify vulnerabilities for ODL and ONOS related to resource consumption, authentication, integrity, confidentiality, and other factors that take advantage of the network management datastore architecture (NMDA)[4] design [79], [80]. The main drawbacks are detailed below:

Regarding ODL, the majority of detected vulnerabilities lead to increased usage of resources and consequently to a DoS [81]–[83]. Likewise, other vulnerabilities allow spoofing actions over the network topology [84], [85]. In terms of integrity and confidentiality in ODL, vulnerabilities have been found to be derived from a Netconf TCP service bug [86]. These vulnerabilities, in conjunction with an XML eXternal entity (XXE) attack, can allow the inclusion of local and remote files [87]. Additionally, the lack of an automatic cache cleaning mechanism after password alteration in the ODL AAA module allows an attacker to exploit this vulnerability to modify files or system information [88]. Even though there is no longer any activity in the Defense4all mechanism, it is important to mention that it has already been broken by remotely authenticated users [89]. Regarding ODL clusters, the main attention must be focused on the message exchange process between instances since the clusters lack encryption and authentication mechanisms [90].

On the other hand, in the ONOS controller, vulnerabilities related to DoS have been identified. One is related to the lack of limits in the allocation of resources and another to the unexpected closure in the OVSDB component [95], [96]. Additionally, exception mismanagement when jumbo frames are registered can lead to a controller shutdown [97]. ONOS presents vulnerabilities affecting confidentiality and integrity related to XXE, which can be exploited by using an OpenConfig Terminal Device or when authentication mechanisms are not used [98], [99]. Additionally, vulnerabilities related to authentication have been identified in the controller user interface, with which it is possible to perform actions on the loading of applications or have access to the information of the network topology [100], [101]. Recently, a vulnerability was discovered that compromises the integrity, confidentiality, and availability of the network due to mishandling backquote characters [102].

In addition to the mentioned attacks, consideration should be given to attacks arising from zero-day vulnerabilities, which can affect all controllers and thus the entire SDN architecture. These attacks can be addressed by implementing IDS. There are two types of IDS: the signature-based intrusion detection system (SIDS) and the anomaly-based intrusion detection system (AIDS) [103]. Of these two,

[3]https://owasp.org/

[4]https://www.rfc-editor.org/rfc/rfc8342.txt

TABLE 3. Summary of contributions for the east-/westbound interface and control plane.

Ref.	Addressed Problems	Main Solution Approaches	Literature Contribution					
			S	T	R	I	D	E
[64]	DDoS attacks	Gossip Protocol					✓	
[72]	Multicontroller communication	ACL method	✓		✓			✓
[73]	Multicontroller communication	DFS, DLBS and SSL	✓	✓	✓		✓	✓
[74]	Multicontroller communication	IBC	✓					✓
[75]	Multicontroller communication	ECC	✓	✓		✓		
[91]	Lack of intrusion and anomaly detection	Machine learning					✓	
[92]	Lack of intrusion and anomaly detection	Deep learning	✓				✓	
[93]	Lack of intrusion and anomaly detection	Deep learning	✓	✓		✓		✓
[94]	Lack of intrusion and anomaly detection	Deep learning					✓	

S=spoofing, T=tampering, R=repudiation, I= information disclosure, D=denial of service, E=elevation of privilege

the SIDS is less efficient in detecting zero-day vulnerabilities because its functionality is reactive. In other words, the attack must be known so that it is included in the signature. The AIDS tries to provide solutions proactively by using methods, such as statistics-based, knowledge-based, and machine/deep learning-based methods [103]. In this regard, the following contributions are cited to provide solutions to SDN security issues:

In [91], the authors propose an IDS named Eunoia, which uses machine learning and is composed of three subsystems: data preprocessing, predictive data modeling, and decision making and response subsystems. The first subsystem is responsible for eliminating irrelevant data through statistics, traffic analysis, and feature engineering. The filtered data are sent to the predictive data modeling subsystem, which uses a learning algorithm called random forest to identify intruders. Simultaneously, the decision making and response subsystem work on imprecise data to achieve greater filtering accuracy.

Similarly, in [92], using deep learning, a framework for anomaly detection is proposed. It consists of two modules: the anomaly detection module and the data delivery module. The first module is responsible for the security part, focusing on several attack vectors (e.g., hijacking, spoofing, malware). This module, through the restricted Boltzmann machine (RBM) and the support vector machine (SVM) algorithm, collects and classifies the characteristics of the flows transmitted between the controller and the network elements to generate an anomaly report. Based on this report, the controller can take action on the forwarding equipment and discard the packet and the communication. The second module is responsible for data delivery quality to provide quality of experience (QoE).

Malik *et al.* [93] present a framework for intrusion detection based on hybrid deep learning techniques, applying long short-term memory (LSTM) and a convolutional neural network (CNN). LSTM avoids the vanishing gradient problem present in large dataset sequences, while CNN takes care of feature extraction from the raw data. This solution is mainly trained for application-type attacks such as port scan, cross-site scripting, and botnet.

In [94], the authors propose an IDS using deep learning. This solution is composed of three modules: flow collector, anomaly detector and anomaly mitigator. In the first module, all the sensitive information of the packet-in messages is obtained. In the anomaly detector module, the anomaly detection process is generated using gated recurrent unit recurrent neural network (GRU-RNN). Finally, the anomaly mitigator module decides whether to discard the traffic or analyze it in depth.

Keeping the same scheme as Table 2, Table 3 summarizes the main contributions regarding the security solutions for the control plane and the east-/westbound interface. Since attacks are usually deployed from the data or application planes, the solutions for various issues that put the controller at risk are visible at these planes.

C. SOUTHBOUND INTERFACE AND DATA PLANE

The data plane and the control plane communicate via the southbound interface using protocols such as OpenFlow [104], OVSDB [105], OpFlex [106], NETCONF [107], and ForCes [108], among others. The protocol most widespread and currently studied is OpenFlow, already considered a standard according to [104]. Therefore, in this paper, an emphasis will be placed on this protocol.

By itself, OpenFlow has no security mechanisms. Secure communication connections can optionally be established between the OF switch and the OF Controller via TLS or plain TCP for the prime connections or through TLS, DTLS, TCP, or UDP for the secondary connections [6], [109]. The ONF recommends TLS from version 1.2 onwards, despite its vulnerabilities [110], [111].

Some research has determined the existence of security problems in the communication channel. In [74] and [112], the authors argue that the lack of use of certificates at handshake time in the authentication process on the client side and the TLS protocol misconfiguration can lead to MiM attacks. On the other hand, Benton *et al.* [113] indicate that the lack of TLS configuration increases the risk in the switches, since by not having authentication and, in many cases, having the "listening mode" active, the attacker could have access to the forwarding information and rules. In this sense, to mitigate the MiM attack in the southbound interface, in [112], the authors propose an extension to the TLS protocol. The authors of [74], [114] employ the IBC protocol to secure communications in the Southbound Interface and the data plane.

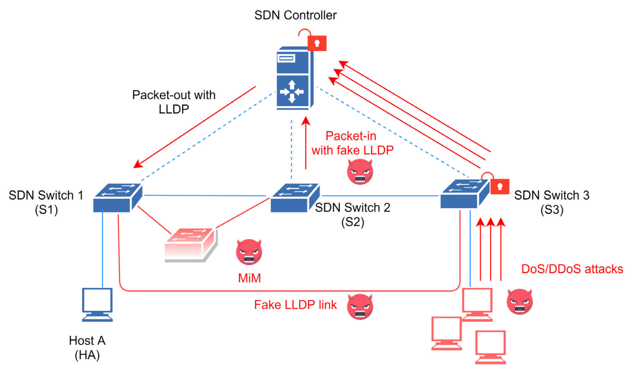

FIGURE 3. **Associated attacks in the data plane and southbound interface.**

Another relevant aspect to consider in the communication channel between the data and control layers is saturation, which could lead to network unavailability. In this context, there are two ways to minimize the signaling overhead between the data and control planes [115]. The first can be the delegation of responsibility and decision-making power to the data plane by transforming it from stateless to stateful, which will be addressed later. The second contemplates the configuration of the controllers and switches so that the flow rules are handled in the proactive mode [109]. Regarding this last option, it should be taken into account that although the proactive mode would be able to reduce traffic between the data and control layer, it could also saturate the memory of the switches; therefore, its use is advisable in environments where there is extensive knowledge of the network requests [116].

The data plane concentrates all the network or forwarding elements, such as switches and routers. The network elements are used to implement all the decisions taken to respond to requests. One of the most important actions involving the data plane is topology discovery, its updating, and forwarding decisions based mainly on two services: the Link Discovery Service (LDS) and the Host Tracking Service (HTS) [117].

LDS generically uses the Link Layer Discovery Protocol (LLDP) to collect switch information and inter-switch link information [118]. In SDN networks that implement OpenFlow, obtaining link information between OF switches occurs via the OpenFlow Discovery Protocol (OFDP). However, collecting link information between OF switches and traditional switches is done by using the Broadcast Domain Discovery Protocol (BDDP) [119]. Both OFDP and BDDP are adaptations of the LLDP protocol. On the other hand, the host tracking service maintains information about the hosts and their positioning within the network.

The process of these services is focused on the switch-controller handshake. This event can be summarized as an exchange of packet-in and packet-out messages. Each controller determines the intervals for these messages exchanges between entities. With the updated topology information, the controller can manage the network resources properly, allowing the re-routing of traffic according to the real needs of the environment, discriminating better paths, if required, and giving a better quality of service (QoS) to the applications provided by the application plane [120].

Despite the importance of both services, it has been observed that the packet exchange process for the topology update presents security problems derived from the lack of security mechanisms in the Host Tracking Service of the controller and the lack of adequate authentication mechanisms for the origin of the LLDP packet, which arrives transformed to the controller into a legitimate packet-in message and therefore is treated as a real requirement. Thus, an attacker (malicious host or switch) can achieve topological poisoning attacks [9], [121]–[123], such as a host location hijacking attack or a link fabrication attack [124], to introduce illegitimate information, to build new routes and thus divert traffic for malicious purposes. In addition, within the process of updating the network topology, other attacks can be triggered, such as DoS (LLDP flooding or packet injection attack) [125], topology tampering attacks (port probing, port amnesia) [126], and the repudiation or MiM, the latter being executed through a silent relay attack [127].

Up to this point, it has been observed that there are several security issues in the network topology discovery process. However, there are other problems that are not exclusive to SDN networks such as DoS and DDoS attacks from the end clients (host). These attacks constitute a great challenge for any infrastructure administrator since their identification and mitigation depend on their deployment. DoS attacks are launched from a single host, so their treatment can be easier, unlike DDoS attacks, which are launched through multiple hosts, usually botnets, and whose identification is more complex. Likewise, it has been observed that DoS attacks can be launched in conjunction with impersonation attacks, such as MAC or IP address spoofing [128], [129], or can even be triggered after an inference attack [130].

To address the mentioned security gaps in the data plane, several authors have proposed different solutions that will be covered in this manuscript. However, to clarify the narrative, this manuscript highlights the difference between stateless data planes and stateful data planes.

In a stateless data plane, the network elements do not store network states. The network elements implement the solutions of the decisions taken in the control plane. All new actions to be performed by the stateless data plane must be queried to the controller [131]. However, the control plane can delegate functions to the data plane whenever appropriate and necessary for "dynamizing" the network behavior. This delegation allows the data plane to store network states and take actions, thus turning a stateless data plane into a stateful data plane [15].

Figure 4 shows the difference between stateless and stateful environments, including an approach related to their security. Thus, the presence of attacks is estimated predominantly in the processing cores in each case. In a stateless environment (Figure 4a), attacks can mainly affect the controller without ruling out the network elements. On the

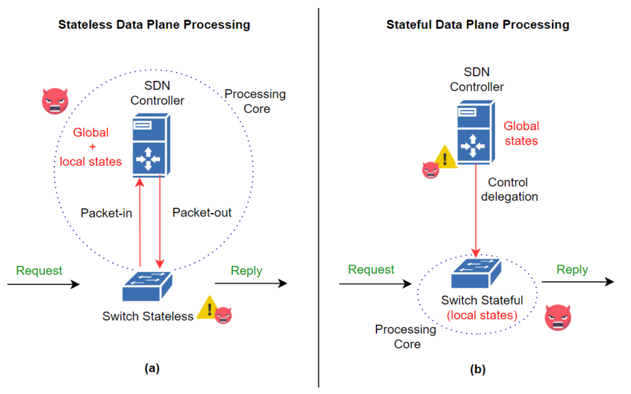

FIGURE 4. Stateless and stateful data plane. Adapted from [132], [133].

other hand, in a stateful environment (Figure 4b), the attacks are more visible in the network elements and can also extend to the controller.

1) STATELESS DATA PLANE

There are several solutions to mitigate or detect attacks on the network topology (hijacking, link fabrication, packet injection attack, etc.); among them are the following.

Dhawan *et al.* [57] present SPHINX as a framework capable of integrating with the controller as a module or as an application. This is a countermeasure for topology violations and DoS attacks. This solution monitors the OpenFlow messages between the controller and the switch to extract the topology and forwarding states of the network. In this way, it builds flow graphs, which are checked against the policies learned or established in the policy engine. Thus, in the case of suspicion of an anomaly in the network, an alarm is raised.

TopoGuard [124] is an extension to avoid a host location hijacking attack and a link fabrication attack on controllers that work with OpenFlow. TopoGuard consists of four main modules. The ports manager monitors the OpenFlow messages of the switch ports. The port property stores information (precondition) used to verify the reliability of the topology update. The host prober is the counterpart in the reliability verification process (postcondition). Finally, the topology update checker performs the topology information verification process based on preconditions and postconditions and validates the origin and integrity of LLDP packets via a keyed-hash message authentication code HMAC/TLV. Additionally, in the case of detecting an anomaly in internal link updates, it blocks the port and raises an alert. In this way, it manages to mitigate network topology poisoning attacks, enabling several open-source controllers to work with this solution, despite this solution being questioned in some scenarios [134].

In [127], the authors indicate that silent relay attacks (MiM-type attacks) that can inject false links between switches may exist in the topology discovery process. In addition, the authors generate a brief review of TopoGuard, noting that despite its implemented controls, it fails to detect silent relay attacks. Given this, a solution named silent relay detector is presented, with a main objective to force the attacker to behave unusually. The solution detects an attacker executing a silent relay between switches by comparing the size of the payload of the LLDP messages it sends with the allowed MTU size.

Similarly, [125] show that TopoGuard cannot control the packet injection attack since its procedure does not verify the legitimacy of the MAC address origin. Therefore, they propose PacketChecker, which is categorized as a module for controllers working with OpenFlow. It has two main parts: attack detection and attack solution. attack detection checks packet-in and extracts information used to associate the switch port with the host MAC address. Thus, if a packet-in arrives with a different MAC address from the same switch port, it is sent to the attack solution for the switch to discard it, as it is considered a malicious message.

INSPECTOR [135] is a hardware-based solution for packet injection attacks. This solution relies on a device added to the network architecture to authenticate the origin of the packet-in messages through a database of valid hosts. Thus, if a host is not authenticated, its packet-in message will be discarded.

TopoGuard+ [126] is an improved version of TopoGuard, in which two modules are added to detect anomalies either in the interactions (control message monitor (CMM)) or in the latencies (link latency inspector (LLI)) during the LLDP packet exchange process. With this, it is possible to mitigate port probing and port amnesia, tampering attacks that managed to evade the TopoGuard and SPHINX mechanisms.

There are also proposals to strengthen the OFDP protocol, such as sOFTDP [123]. The main difference between OFDP and sOFTDP is that unlike OFDP, which uses clear MAC addresses, sOFTDP uses the hash values of MAC addresses in the LLDP exchange process. sOFTDP works with a bidirectional forwarding detection (BFD) mechanism in asynchronous mode to obtain connectivity information. sOFTDP maintains topology memory, with a database of links to choose the shortest route for forwarding. sOFTDP has fast-failover groups, which check the switch port status and, if required, generate a change to a backup link. Finally, it has LLDP drop rules, which allow it to eliminate harmful LLDP packets and hashed LLDP content for sending encrypted LLDP packets.

For conducted attacks that target service availability, there are solutions based on traffic analysis statistics within a given time frame, as well as those that rely on machine learning. Although these solutions do not reduce the number of attacks, they can help to detect and halt them early to undermine the harmful impact on the network services. The following solutions are available.

DAISY [136] is a proposal for detecting and mitigating DoS attacks. DAISY has four main functions. The data collection function collects and stores information from - in messages. Threat detection, through statistics, determines whether there is an excess of requests from the host. If there is an excess, this traffic is classified as suspicious, and the attack prevention function blocks this traffic for a short time.

If the host continues sending requests in the same amount, the traffic is considered malicious and is blocked for a longer time. Finally, the threat value reduction function updates the blocking flow rules after each system iteration.

FloodGuard [137] is a security solution that deals with a DoS attack known as data-to-control plane saturation. It uses two OF modules included as an application in the controller: the proactive flow rule analyzer and migration modules. The latter is composed of two submodules: the migration agent and the data plane cache. When through the migration agent, FloodGuard detects an attack, it redirects the table-miss to the data plane cache, and the system goes into defense mode, sending the benign flows to the controller without reloading it. In parallel, the proactive flow rule analyzer generates and installs the new flow rules directly on the data plane and keeps them dynamically updated. When the end of the attack is detected, the data plane cache stops receiving flows and returns to a normal state.

In [138], the authors formulate an entropy-based solution using a security gateway and a HoneyPot. The security gateway, through defense and filtering algorithms, determines whether there is a DDoS attack. If the attack exists, the s are sent to the HoneyPot. Otherwise, forward rules are requested to the controller to deploy them in the switches.

In [139], the authors expose a saturation attack called a table-miss striking attack. This attack occurs when a malicious entity learns sensitive and confidential information from the control plane. With this information, the attacker can generate traffic patterns and therefore enough communication in the control-data plane to saturate it. In response to this, Xu *et al.* [140] propose a solution named SDNGuardian. It has four modules: preprocessor, attack detector, traffic filtering, and rule sweeper. The preprocessor and the attack detector modules are responsible for the identification, extraction, and storage of the sensitive fields of a -in. With the extracted information and using entropy algorithms, a determination is made as to whether an attack exists. The last two modules are responsible for identifying the switch ports under attack to limit their speed and remove the malicious rules from flow tables, thus containing the attack.

SDNManager [141], an approach for DoS attack detection and mitigation, consists of five parts. The monitor collects network states and converts them into variables. The variables are sent to the forecast engine module to generate a statistically based bandwidth consumption forecast. The checker module validates whether the bandwidth used is equal to the forecasted bandwidth. The updater and storage service modules update the network and store the variables, respectively. When it detects abnormal traffic, this solution generates a penalty on bandwidth consumption, giving an attacker lower priority.

Sahoo *et al.* [142] propose a framework to detect and mitigate DDoS attacks using machine learning. This solution consists of several modules. On the one hand, the statistics monitor module receives the flow statistics information from the switches at certain time intervals. Subsequently, the feature extractor module obtains the flow characteristics using kernel principal component analysis (KPCA). The extracted features are used in the classifier module, which works with the SVM classifier with parameter optimization by using the genetic algorithm (GA), thus identifying malicious traffic from benign traffic. With this information, the mitigation module can create a rule for the underlying network to discard malicious s.

Aujla *et al.* [143] propose a blockchain-as-a-service architecture to mitigate DDoS attacks. The authors detail their proposal procedure, which is summarized as the identity validation of a switch requesting access to the network before the start of flow transmission. In this solution, the public and private keys are generated via blockchain and shared between the network switches. Therefore, each time a device generates a request on the network and after a consensus of the other members, transactions are accepted or rejected.

Since botnets often use P2P communication, in [144], the authors propose and test a programmable module with machine learning to identify malicious P2P traffic. This proposal is composed of three essential parts: a rule arbitrator (controller), data-link bridges (OpenFlow switches), and a detection agent. After a process of incoming duplication and flow recognition, this solution allows to label each flow as a P2P or a benign P2P application through learning models. Once this information is available, a modification is made to the flow tables, thus discarding malicious s.

Additionally, it has been observed that many botnets take advantage of HTTP features to generate DDoS attacks that are expressed through HTTP GET flood attacks. In this sense, the authors of [145] propose a solution that combines hardware and an entropy mechanism. The mechanism called per-URL counting works with two sections, namely, one for detection and one for mitigation. In the detection section, the HTTP GET request is validated through several filters, one of which is the counting filter. If the counter exceeds a threshold, it is considered an attack, and the host is placed on a blacklist, which is handled by the mitigation section. Finally, the entire system is presented on FPGA hardware.

In [146]–[148], for DDoS attacks, early detection solutions using entropy are presented. However, the authors of [149] argue that entropy measures for DDoS attacks work when the attack target has a fixed IP address, but when the attack is to a random IP, the mitigation measures are limited; since the thresholds do not consider the possible variances, they propose a solution with principal component analysis (PCA), which from the information collected, provides new models that allow predicting the attack. Similarly, regarding randomly targeted DDoS attacks, an early detection solution is proposed in [150]. The solution consists of three phases. In the first one, information is collected from the switches. The second uses algorithms that calculate thresholds using the EWMA statistical model, with which it is feasible to handle the volatility of the dataset. In the third phase, attack detection is performed by comparing the values obtained from the previous phase with the table miss of the switches.

Likewise, in terms of the early detection of DDoS, IDS countermeasures have been proposed, as in [151], a paper in which the authors use IDS Snort rules[5] and generate alarms in the event of detecting an attack.

In [152], the authors present an IDS with machine learning algorithms to detect DDoS attacks. The solution is composed of two modules. The first classifies normal or anomalous host behaviors through signatures. If anomalies associated with a host are detected, this is delegated to the second module, which will determine the legitimacy of the host through a three-way handshake. In case this process is not completed, the host can be included as an attacker in an access control list.

For detecting and defending against DDoS attacks, Li *et al.* [153] propose a method that employs deep learning. This solution works with several modules responsible for the extraction of characteristics of switch s to subsequently obtain statistics of repeated patterns through which the existence or absence of a DDoS attack can be determined.

In [154], a solution for detecting low and high traffic volume DDoS attacks is presented. This solution has two types of thresholds, namely, static through entropy and dynamic through machine learning. In this proposal, the incoming traffic is used to train the classifiers.

Wang and Chen [128] propose a solution called SGuard to mitigate spoofing and DoS attacks. It is comprised of three modules: access control, classification, and a data plane cache. Access control collects information from network users and checks it against an SGuard ACL, preventing spoofing attacks. The classification module distinguishes anomalous traffic from benign traffic by employing machine learning (self-organizing maps (SOMs)). The data plane cache module is responsible for maintaining the table-miss s during an attack, thus avoiding resource exhaustion in the data plane.

In [155], the authors address two types of attacks: spoofing route attacks and resource exhaustion attacks. The former deals with a module installed on the OF switch named ‘‘selective blocking’’; this module extracts information (IP, MAC) when it receives a new request to the network. This information is compared against a pre-existing dataset. In the case of a duplicated outcome, the host, considered impersonalized, is blocked and is treated as malicious. In the second attack, the authors use a detection and prevention module called periodic monitoring. This module validates the amount of traffic sent in a specific time. If the values exceed a threshold, an alarm is triggered, alerting the controller for an eventual blocking event.

In OpenFlow-based SDN networks, switch flow tables can be susceptible to inference attacks [130]. In this type of attack, employing data mining and different algorithms, adversaries can estimate, among other things, the size of the flow table of switches. With this information, the attacker can generate requests that saturate the memory of the switches, leading to a DoS. Although the authors of [156] consider that deploying inference attacks in a real network would be extremely complex, they propose two countermeasures to prevent such attacks. The first is based on the randomization of network attributes technique [157], and the second makes use of Openflow rate-limiting combined with proactive rules to mitigate inferred ICMP attacks or rate-limiting with a proxy to mitigate TCP attacks.

Even with the mentioned countermeasures, there may be overloads in the flow tables, which may or may not come from malicious action that can negatively influence the availability of the network; therefore, it is necessary to maintain proper management of the switches' memories so that they respond correctly.

According to [115], there are three ways to manage switch memory efficiently: rule eviction, rule compression, and rule split and distribution. The former, which is the most used, replaces old entries in the flow tables with new entries. It leverages the use of caching replacement algorithms (e.g., least recently used (LRU) and first-in-first-out(FIFO)), on the flow state and on the hard timeout/idle timeout at the switches. The second way, the rule compression/aggregation, seeks to reduce the number of rules by using wildcards to fit the flow table. Compression can be performed on both the access control rules and the forwarding rules. Finally, the latter propagates the rules among several switches [115]. In the literature, several approaches that use switch memory management techniques have been found [158], [159]. However, there are few oriented to a security scope; thus, in [160], the authors propose a detection and defense mechanism against LDoS (low-rate DoS) attacks. The proposal uses statistical analysis for detection, whereas for mitigation, it uses the replacement technique supported by the LRU algorithm. In [130], an inference attack model and two countermeasures are proposed. The attack model uses FIFO and LRU replacement algorithms. Regarding the countermeasures, the first is based on routing aggregation, and the second is based on a flow table architecture with two levels, level one consisting of TCAMs and level 2 consisting of SRAMs.

On the other hand, for impersonation attacks, protocols, such as EAP, EAPoL, and RADIUS, are used to authenticate and authorize network clients before they access the network. Under this premise, solutions have been proposed, such as [161] and [162], which work with the EAP protocol in reactive mode, or [163], which achieves its goal through EAPoL-in-EAPoL encapsulation in proactive mode.

2) STATEFUL DATA PLANE

Most programming languages for SDN have been based on OpenFlow 1.0, which has a stateless data plane configuration by default. Thus, the switches only comply with the forwarding rules that were issued by the controller. This process sometimes generates overhead for the controller and latency to answer the need for network abstractions, such as firewalls and load balancers [172].

[5] https://www.snort.org/

TABLE 4. Summary of contributions for the southbound interface and data plane.

Ref.	Addressed Problems	Main Solution Approaches	Literature Contribution					
			S	T	R	I	D	E
[112]	MiM attack communication channel	TLS protocol extension		✓			✓	
[74], [114]	MiM attack communication channel	IBC protocol		✓			✓	
[124]	Network topology information poisoning (Host location hijacking attack and link fabrication attack)	Preconditions/ Postconditions	✓	✓			✓	
[57]	Network topology violation and DoS attacks	Flow Graphs		✓			✓	
[127]	Fake link injection	LLDP payload size check		✓			✓	
[126]	Network topology information poisoning (Port probing and port amnesia)	Preconditions/ Postconditions	✓	✓			✓	
[123]	OFDP vulnerable	BFD	✓	✓		✓	✓	
[125]	Fake packet-in	Switch port association with host MAC	✓				✓	
[135]	Lack of packet-in message authentication	Independent hardware implementation	✓					
[136]	DoS attacks	Statistics					✓	
[137]	DoS attacks	Protocol-independent defense framework					✓	
[128]	Spoofing and DoS attacks	ACL / Machine learning	✓				✓	✓
[155]	Spoofing Route and Dos attacks	Traffic statistics	✓				✓	
[138]	DDoS attacks	Entropy					✓	
[140]	DoS attacks	Entropy					✓	
[141]	DoS attacks	Traffic statistics					✓	
[142]	DDoS attacks	KPCA+GA+ Machine learning					✓	
[143]	DDoS attacks	Blockchain					✓	✓
[144]	Lack of P2P traffic identification	Machine learning					✓	
[145]	HTTP DDoS attacks	Entropy + Hardware					✓	
[149]	DDoS attacks	PCA					✓	
[150]	DDoS attacks	EWMA					✓	
[151]	DDoS attacks	Snort IDS					✓	
[152]	DDoS attacks	Machine learning					✓	
[153]	DDoS attacks	Deep Learning					✓	
[154]	DDoS attacks	Entropy / Machine learning					✓	
[156]	Inference attacks	Randomization of network attributes/ Rate-limiting + Proactive rules Rate-limiting + Proxy					✓	
[160]	DoS attacks (LDoS)	Statistics / LRU					✓	
[130]	Inference attacks	Routing aggregation / TCAM + SRAM					✓	
[161], [162]	Lack of network client access control	EAP / RADIUS						✓
[163]	Lack of network client access control	EAPoL / RADIUS	✓					✓
[164]	DoS attacks	Blockchain + Hardware					✓	
[129]	SYN flooding and ARP spoofing attacks	SYN/ACK and ACK/FIN packets' ratio / P4 cache	✓				✓	
[165]	Traffic overload / Latency	App+P4					✓	
[166]	Traffic overload	Snort IPS + P4					✓	
[167]	DDoS attacks	P4+Entropy+FSM					✓	
[168]	Lack of link protection between stateful switches	MACsec	✓	✓		✓		✓
[169]	States exchange between stateful switches	Digital signatures	✓					✓
[170]	Link flooding attacks stateful data plane	Topology obfuscation					✓	
[171]	DDoS attacks	Machine learning + P4					✓	

S=spoofing, T=tampering, R=repudiation, I= information disclosure, D=denial of service, E=elevation of privilege

In this sense, it is noted that it is possible to extend the programmability of network applications to the data plane, keeping the local state information in the switches so that they can take control of packet forwarding without querying the controller (stateful data plane), providing "greater dynamism" to the network.

To provide programmability to the data plane, some of the proposals are the following: SDPA [172], which proposes a "match-state-action" abstraction instead of OpenFlow's "match-action" processing; SNAP [173], which bases its programmability on an abstract network and persistent global arrays; and those proposals based on XFSM tables (eXtensible Finite State Machines) for flow processing in switches, such as FAST [174], OpenState [175], OPP [176], and the best-known P4 [177].

Supported by the programmability solutions described above, multiple network abstractions have been developed in a Stateful Data Plane, including firewalls [178], [179], traffic management applications [180], and load balancers [181], [182], among others [183].

Similarly, under this trend, in [164], the authors propose the blockchain-enabled packet parser (BPP) architecture for detecting up to five attack categories, the most representative of which is DoS. This solution merges blockchain and P4 and is implemented on FPGA. P4 customizes packet processing in the data plane, while blockchain examines packet behavior.

In the case of attack detection, the controller is reported, and actions are taken under defined policies.

Lin *et al.* [129] studied SYN flooding and Address Resolution Protocol (ARP) spoofing attacks. For the former, the authors use the SYN/ACK and ACK/FIN packets' ratio. Thus, if the ratio between these packets does not match, the network has anomalous traffic. Regarding the latter attack, the authors manage to discard repeated ARP packets by using the P4 cache to reduce the load on the controller.

In [165], a P4-based security framework to reduce traffic overhead and latency is proposed. It is composed of two primary parts: a StateFit App and a StateFit Interpreter. The former is installed as an application on the controller and is responsible, among other things, for traffic analysis. The latter acts on the P4 switches. Here, traffic is processed and filtered according to established policies.

In [166], the authors propose P4ID, a solution that seeks to reduce network traffic by generating filtering on P4 switches before packets are sent to an IDS. P4ID is composed of two main parts: the rule parser and P4 implementation. The former works with the Snort IPS rules. These rules are installed on the P4 switches so that the initial match action can be performed organically. Through P4, the authors can process stateless and stateful packets.

Ilha *et al.* [167] present a proposal to detect and mitigate DDoS attacks in stateful environments with P4. For detection, the authors use entropy, whereas for mitigation, they use finite-state machine FSM.

Despite the advantages of a stateful data plane for traffic reduction between the data and control planes during the processing of requests, it has been observed that this type of environment could jeopardize the consistency of the network [184]. The reason behind this is the lack of an authentication mechanism between the switches, making them prone to impersonation by malicious agents at any time and giving rise to attacks. Therefore, a proposal for secure link discovery was presented to handle LLDP packets protected through encryption, decryption, and authentication. The proposal, called P4-MACsec [168], protects links between P4-based switches by using the IEEE 802.1AE standard (MACsec). This solution contains three main parts: Packet switching with MAC Address Learning, Secure Link Discovery, and Automated Deployment of MACsec.

Similarly, the authors of [169], concerned about state exchange between P4 switches, propose an authentication solution using digital signatures. This solution creates a hash chain attached to each packet that is transmitted for state exchange. Although the public key controller is responsible for the final part of the chain, the verification operations are performed in the data plane.

In [170], a solution for link flooding attacks on programmable data planes is presented. The idea of this proposal is to route malicious traffic through available paths, employing topology obfuscation. In this work, the attacker believes that his flow is achieving its malicious purpose; however, it is discarded.

The authors of [171] suggest a measure for DDoS attack detection that uses machine learning and relies on P4. Through the switches, traffic information is obtained, which is classified with algorithms such as K-nearest neighbors (KNN), random forest, and SVM to determine whether there is an attack.

Even with the solutions described above, there are still exploitable vulnerabilities in a stateful data plane. For instance, this data plane must deal with all the data that a controller manages in a stateless data plane scenario. This task could eventually saturate the switches' TCAM memory and increase its CPU consumption until the point of self-denial of service [10].

Table 4 summarizes the solutions to the problems presented in the southbound interface and data plane, distinguishing the solutions for stateless data plane and for stateful data plane.

III. DISCUSSION, OPEN CHALLENGES, AND FUTURES DIRECTIONS

Existing studies have proposed different solutions for various scenarios and security issues of SDN architecture. Nevertheless, security remains a challenging research area with many unresolved questions. In this section, we will discuss some open issues and future directions that may merit research attention.

A. REGARDING SDN PLANES

In the application plane of SDN, many of the authors address authorization issues through fine-grained access control solutions, as they agree that due to the dynamic behavior of the architecture, granularity concerning permissions is mandatory [43], [46], [47]. Likewise, since the main issue is the trust relationship between the applications and the controller, there is a research opportunity regarding the generation of a standardized repository that validates the security levels of SDN applications. As a practical example, it is possible to cite the HP SDN app store, which in 2014 launched a series of applications with which users can interact [185].

At the control plane, there is a vast range of controllers. These have been either introduced by academia or by the industry [56], [186]. Some of them have been assessed to check their effectiveness under particular attack scenarios, whereas others are still under study [76], [80]. In the control plane, it is possible to appreciate the differences between centralized controllers and distributed controllers. Specifically, the fault tolerance mechanisms that the latter implements are able to reestablish the service after a system failure (e.g., DoS attack). However, many authors argue that there is a crossroad between resilience and consistency [186]–[190]. This is because, according to Brewer's theorem, or the CAP (consistency, availability, partition tolerance) theorem, in distributed environments, after a partition, availability and consistency cannot be equally assured. Although there are controllers such as ONOS and ODL that implement the RAFT algorithm as a fault tolerance or strong-consistency mechanism, there are still some gaps to address regarding improving their performance. This issue is relevant mainly for providing good availability without sacrificing the information between controllers after recovering from a service interruption. To the best of our knowledge, only a few authors have taken action on this point from a security approach; for instance, to improve the RAFT algorithm against DDoS attacks, Hanmer *et al.* [191] present and assess a proposal called BabbleResistantRaft.

The data plane no longer presents itself solely as a set of forwarding elements. Depending on the needs, the data plane may respond to requirements without constantly querying the controller. Whether enabling the configuration for proactive flow rule handling or delegating functions to the data plane, a dilemma is created between reducing traffic on the communication channel (control-data planes) and potentially overloading the switches. In this regard, there are techniques for managing the switches' memory, which should be well analyzed before their use. The authors of [115] argue that the compression/aggregation technique has limitations in its use when working with certain versions of OpenFlow since not all match fields can work with wildcards. However, some studies that perform compression with OpenFlow v1.3 obtain a reduction in the size of the flow tables up to 60% [192]. It should also be considered that with compression, there may be a risk associated with information disclosure, even though authors such as Rifai *et al.* [193] affirm that their proposal can deal with this type of inconvenience. Finally, before using compression, it is necessary to evaluate the processing capacities, since not having them may generate a computational overload and eventually a self-denial of service. Likewise, in the Split and distribution technique, there is a risk to the flow rules, since when they are propagated, they could be lost, duplicated, or could overload a node. Regarding the eviction technique, the LRU and FIFO algorithms have been tested, revealing that FIFO shows better performance and resilience to DoS attacks [194].

Recently, much focus has been given to the delegation of functions from the control to the data plane (stateful data plane), although this can lead to losing the original context of SDNs. Since decisions are handled at the switch level and there is no control entity with a permanent review, inconsistencies may arise in the network topology [10]. Despite the considerations presented by stateful data planes, data plane programmability options such as P4 are beginning to gain momentum in solving security problems. Simultaneously, FPGA functionalities are being leveraged, as they bring programmability and processing agility whereby latency and jitter can be reduced in the network [195], [196]. On the other hand, although this manuscript classifies security solutions in a layer- or interface-focused way, there are solutions that explore the entire SDN architecture. For instance, in [197], the authors validate inconsistencies in network flow policies. In [198], the authors use a fuzz testing algorithm to assess SDN environments. Frameworks for detecting and remediating attacks, such as DoS/DDoS, scanning, or intrusion at different levels of the SDN architecture, are presented in [199], [200]. The authors of [201] perform SDN domain control by enforcing ACLs according to specific policies.

B. REGARDING SDN INTERFACES

Most of the authors' attention is focused on exposing and solving attacks in the various planes of the SDN architecture, leaving aside the interfaces' security issues, which should be a point of reflection for both academia and industry. The north and east/west interfaces represent an important part of the large-scale deployment of SDN since they allow interoperability and access to a wide variety of applications. However, in the absence of standardized interfaces, during integration between TOs, the likelihood of attack scenarios increases. Therefore, one of the main challenges to be considered for the security of SDN architecture is the standardization of interfaces. The advantages of interface standardization in terms of security are reflected in resource optimization. On the one hand, it would reduce the number of attack fronts that currently exist due to the wide variety of third-party applications. Second, the research lines would be oriented towards reinforcing the architecture's protection measures under a regulated scheme. Finally, in the case of an imminent attack, the mitigation efforts would be unified, and solutions would be found early on.

Regarding the south interface, the ONF recommends employing TLS from version 1.2 onwards to secure communication in OpenFlow implementations; however, this is a vulnerable version [110], [111]. Thus, now, the use of version 1.3 would be the most suitable.

The literature shows that TLS certificate configurations with RSA are the most common for securing communication channels [202]. However, there are other algorithms that could be used, such as NTRU or ECC, depending on the deployment. Thus, it has been observed that NTRU has high performance in smaller-scale environments [53]. However, ECC, given its shorter key length, works faster [75].

C. REGARDING THE SOLUTIONS' MECHANISMS

During the development of this manuscript, it has been observed that several of the solutions for attack detection contemplate the use of entropy, a widely adopted mechanism, mainly due to the ease and acceptable cost of its implementation. However, due to the constant evolution of SDN networks, some authors question this mechanism mainly regarding its effectiveness for the early detection of DoS attacks [149], arguing that rigid thresholds could discard benign traffic or worse still accept malicious flow. In this sense, a transition from entropy usage to other mathematical and statistical models, such as PCA or EWMA, is being noticed. Machine learning and deep learning have also been used to propose security countermeasures. These technologies indicate that they are capable of achieving up to 99.98% accuracy in detecting or mitigating an attack [93]. However, their effectiveness depends mostly on the datasets and classifiers used. That is why one of the main challenges regarding the use of machine learning- and deep learning-based solutions for SDN security problems is the training of the classifiers [203].

Blockchain has recently been used to provide security solutions to the SDN architecture in a decentralized manner. Immutability is one of the main features of this technology. This characteristic is especially relevant in environments that want to keep a traceable record of the actions performed in the network. However, the challenges associated with its use should also be considered, namely, the increase in computational cost or possible increases in data processing [143].

At the same time, economic and technical performance should be considered for new contributions, since the existence of an additional layer of security, while mostly guaranteeing that the services provided are protected, can also have an impact on resources and the desired cost-effectiveness [204], [205]. Feng *et al.* [206] focused their attention on security combined with cost-effectiveness and proposed the use of an algorithm called BAGUETTE to halt multidomain controller attacks with a minimum cost, having as one of their future challenges to improve the performance of the algorithm with spine-leaf and full-mesh architectures.

D. REGARDING MANAGEMENT

In recent years, efforts have been made by both academia and industry to counter vulnerabilities and attacks in SDN through technological solutions. Despite this, there are faults directly related to management; many have already been covered in [207], [208]. Nevertheless, this section summarizes the problems of the lack of management in security-related issues.

1) CONFIGURATION

Although both REST and OpenFlow, the two most recognized solutions for handling communication between SDN layers, offer the possibility of lifting the TLS protocol, errors occur precisely in its configuration. Given its complexity, many operators dispense with the use of TLS in switches [6], or in some controllers, the security configuration in HTTP headers is omitted [209]. Noting these drawbacks, there are solutions to reducing the workload for administrators and to provide them solutions for enabling or disabling TLS [210].

2) MAINTENANCE AND TROUBLESHOOTING

Authentication solutions are increasingly oriented towards the use of fine-grained controls. Nonetheless, there are still authors who propose the use of ACLs to assign permissions [72], [128]. Unfortunately, their proposals do not specify the debugging management of the generated lists, which could lead to further complications at the security level. Given the interaction in multidomain, multitenant, and multiservice environments and the incremental deployment of network segments, the total or partial revocation of assigned permissions can be ignored, leaving the network exposed under internal consent.

Likewise, attack scenarios usually occur when there are no updated versions or security patches for the applications [211]. This is not uncommon in the world of SDNs, which demonstrates a lack of maintenance in applications or software that reuses components that had previously presented security faults. In turn, this situation is linked to the insufficient use of knowledge bases, where security incidents are recorded, to minimize response times in the case of faults.

3) FORENSIC ANALYSIS

Security management has a previous phase to minimize vulnerability or prevent/contain an attack. However, once malicious actions have been executed, it is necessary to determine who is responsible and take action over the issue. Considering that a malicious administrator can assign himself read, write and execute permissions, even the ability to delete logs, it is necessary to have tools that support the execution of forensic analysis. Additionally, given the high transactional nature of SDN, due to the decoupling of the layers, there is a problem with the storage of logs, which has an impact on network resources [212]. Noting both the difficulty of log preservation and log manipulation, new solutions for SDN have been proposed [213], [214]. However, it is still necessary to collect evidence of its performance in real environments with a higher number of transactions.

4) AWARENESS

The introduction of any new technology faces the challenge of market adoption. The market could be understood as all network service administrators. Regardless of whether the adopters comprise network administrators, TOs, or smaller data center personnel, training is necessary before implementation, which includes the concept of security as a relevant factor. As a model, we can cite Cisco, which started with certification schools decades ago for both network architecture and security.

5) ASSESSMENT

SDN environments are not free of errors. Therefore, it is necessary to have mechanisms that allow their constant evaluation. Thus, a challenge would be the incorporation of frameworks that handle Deming cycles [215] through the Information Security Management Systems that allow the assessment of an SDN network under the parameters of international security standards, such as ISO 27002 [216].

E. PARTICIPATION WITH OTHER TECHNOLOGIES

Despite the great security challenges presented by the SDN architecture itself, its use by other paradigms and technologies to address security problems is not ruled out. In general, it has been noted that several authors highlight the value of the centralized control that SDN possesses to become an ally in the detection of attacks. For this reason, there are security solutions that make use of SDN to improve their defense mechanisms, while others opt for the joint participation of Network Functions Virtualization (NFV) and SDN to protect against malicious actions in a world where devices are heterogeneous, as in the case of IoE environments or cloud security [217]–[219]. Blockchain has also considered working with SDN either to combine the above paradigms and technologies [220]–[222] or to create an SDN trilogy-Blockchain and OpenStack to achieve secure resource and service sharing outcomes, which could be leveraged in multi-vendor environments [223].

IV. CONCLUSION

In this manuscript, as summarized under the STRIDE categories, the main security problems of SDN architecture and several solutions have been reviewed. The categories most frequently referred to by the authors in their solutions are those related to authentication, denial of service, and authorization. Various authors propose solutions beyond the problems they initially described, generating contributions to several categories of the above methodology. For example, there are authentication solutions that also include authorization or vice versa.

It is important to note that efforts to mitigate attacks are focused on the SDN architecture planes. Nevertheless, standardization of interfaces could strengthen the SDN architecture and thus reduce attack fronts. Additionally, a lack of standardization can affect the use and organic growth of SDN, as the development of APIs without proper guidelines can turn to particular interests or those of a dominant vendor.

Regarding the mechanisms used in problem solving, there is an orientation towards the implementation of fine-grained controls, including as allies machine learning, deep learning and, recently, blockchain. However, similarly for any new technology, it is still necessary to specify details to obtain better results. In this paper, the open challenges to improve the security of SDNs have been exposed.

Finally, this paper has presented a discussion regarding security for SDN architecture, revealing open challenges and future directions. In this sense, it has been determined that it is necessary to generate new fault tolerance mechanisms to ensure consistency and availability, to standardize communication interfaces between layers, to improve proposals for early detection, and to work on new proposals for security management in SDN networks. All these efforts might be complemented by carrying out quantitative studies to evaluate the architecture model based on parameters, such as packet loss rate, latency, and QoE. However, future models must also pay attention to their cost-effectiveness. Note that many of the current SDN security implementations often fail because they cannot be economically or technically leveraged.

APPENDIX

The acronyms used in this article are listed in Table 5.

TABLE 5. Acronyms.

AAA	Authentication, Authorization and Accounting	LSTM	Long Short-Term Memory
ACL	Access Control List	MAC	Mandatory Access Control
AIDS	Anomaly-based Intrusion Detection System	MAC	Media Access Control Address
API	Application Programming Interface	MiM	Man-in-the-Middle
ARP	Address Resolution Protocol	MTU	Maximum Transfer Unit
BDDP	Broadcast Domain Discovery Protocol	NAT	Network Address Translation
BGP	Border Gateway Protocol	NBI	Northbound Interface
CAP	Consistency, Availability, Partition Tolerance	NFV	Network Functions Virtualization
CAPEX	Capital Expenditure	NMDA	Network Management Datastore Architecture
CFG	Control-Flow Graph	NSS	Network Security Services
CNN	Convolutional Neural Network	ODL	OpenDaylight
CSP	Cloud Service Providers	OF	OpenFlow
DAC	Discretionary Access Control	OFDP	OpenFlow Discovery Protocol
DDoS	Distributed Denial of Service	ONF	Open Networking Foundation
DFS	Distributed Firewall Service	ONOS	Open Network Operating System
DLBS	Distributed Load Balancer Service	OPEX	Operational Expenditures
DNS	Domain Name System	P2P	Peer-to-Peer
DoS	Denial of Service	PCA	Principal Component Analysis (PCA)
DP	Data Plane	RADIUS	Remote Authentication Dial-In User Service
DTLS	Datagram Transport Layer Security	RBAC	Role-Based Access Control
EAP	Extensible Authentication Protocol	RBM	Restricted Boltzmann Machine
EAPoL	Extensible Authentication Protocol over LAN	REST	Representational State Transfer
ECC	Elliptic Curve Cryptography	RNN	Recurrent Neural Networks
EWBI	East/Westbound Interface	SBI	Southbound Interface
EWMA	Exponentially Weighted Moving Average	SDN	Software-Defined Networking
FIFO	First-In First-Out	SIDS	Signature-based Intrusion Detection System
FPGA	Field-Programmable Gate Array	SRAM	Static Random Access Memory
GA	Genetic Algorithm	SSBG	Security-Sensitive Behavior Graphs
HTS	Host Tracking Service	SSL	Secure Socket Layer
IBC	Identity-Based Cryptography	SVM	Support Vector Machine
IDS	Intrusion Detection System	TCAM	Ternary Content-Addressable Memory
IETF	Internet Engineering Task Force	TCP	Transmission Control Protocol
IoE	Internet of Everything	TLS	Transport Layer Security
IPS	Intrusion Prevention System	TO	Telecommunications Operator
KPCA	Kernel Principal Component Analysis	UDP	User Datagram Protocol
LDS	Link Discovery Service	WAN	Wide Area Network
LLDP	Link Layer Discovery Protocol	XFSM	eXtensible Finite State Machines
LRU	Least Recent Used	XXE	XML External Entity

REFERENCES

[1] *White Paper 5G Network Technology Architecture*, IGP Group, Chennai, India, 2015.

[2] Z. Lv and N. Kumar, "Software defined solutions for sensors in 6G/IoE," *Comput. Commun.*, vol. 153, pp. 42–47, Mar. 2020.

[3] BusinessWire. (2020). *Global Software-Defined Networking Market (2020 to 2025)—Software-Defined Networking for 5G Presents Opportunities*. [Online]. Available: https://www.businesswire.com/news/home/20200817005303/en/Global-Software-Defined-Networking-Market-2020-to-2025—Software-Defined-Networking-for-5G-Presents-Opportunities—ResearchAndMarkets.com

[4] (2018). *Software-Defined Networking (SDN) Definition-Open Networking Foundation*. [Online]. Available: https://www.opennetworking.org/sdn-definition/

[5] CISCO. (2019). *Software-Defined Networking (SDN) Definition—Cisco*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/software-defined-networking/overview.html

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

[7] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[8] J. H. Cox, J. Chung, S. Donovan, J. Ivey, R. J. Clark, G. Riley, and H. L. Owen, "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25487–25526, 2017.

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

[10] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful SDN data planes," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1701–1725, Mar. 2017.

[11] T. Han, S. R. U. Jan, Z. Tan, M. Usman, M. A. Jan, R. Khan, and Y. Xu, "A comprehensive survey of security threats and their mitigation techniques for next-generation SDN controllers," in *Concurrency Computing*, vol. 32, no. 16. Hoboken, NJ, USA: Wiley, 2020, p. e5300, doi: 10.1002/cpe.5300.

[12] J. C. Correa Chica, J. C. Imbachi, and J. F. Botero Vega, "Security in SDN: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 159, Jun. 2020, Art. no. 102595.

[13] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A survey on security and privacy of 5G technologies: Potential solutions, recent advancements, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 196–248, 1st Quart., 2020.

[14] S. Ahmad and A. H. Mir, "Scalability, consistency, reliability and security in SDN controllers: A survey of diverse SDN controllers," *J. Netw. Syst. Manage.*, vol. 29, no. 1, Jan. 2021, Art. no. 9.

[15] ON Foundation. (2014). *SDN Architecture*. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[16] A. Danping, M. Pourzandi, S. Scott-Hayward, H. Song, M. Winandy, and Z. Dacheng. (Jul. 2016). *Threat Analysis for the SDN Architecture*. [Online]. Available: https://www.opennetworking.org

[17] Open Networking Foundation. (2015). *Principles and Practices for Securing Software-Defined Networks*. [Online]. Available: https://www.opennetworking.org

[18] K. Bissell and L. Ponemon. (2019). *Ninth Annual Cost of Cybercrime Study Unlocking the Value of Improved Cybersecurity Protection*. [Online]. Available: https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf#zoom=50

[19] Microsoft. (2020). *The STRIDE Threat Model: Microsoft Docs*. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN

[20] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[21] (2015). *Open Networking Foundation and ON Lab to Merge to Accelerate Adoption of SDN—Open Networking Foundation*. [Online]. Available: https://opennetworking.org/news-and-events/press-releases/open-networking-foundation-and-on-lab-to-merge-to-accelerate-adoption-of-sdn/

[22] P. V. Tijare and D. Vasudevan, "The northbound APIs of software defined networks," *Int. J. Eng. Sci. Res. Technol.*, vol. 5, pp. 501–513, Jan. 2019. [Online]. Available: http://www.ijesrt.com

[23] S. Raza and D. Lenrow, "Open networking foundation north bound interface working group charter," ONF, Tech. Rep., 2013.

[24] S. Y. Zhu, S. Scott-Hayward, L. Jacquin, and R. Hill, *Guide to SecurITY SDN 78 NFV: Challenges, Opportunities, Application*. Cham, Switzerland: Springer, 2017.

[25] P. Ahmad, S. Jacob, and R. Khondoker, "Security analysis of SDN applications for big data," in *SDN and NFV Security* (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 39–55.

[26] D. Artmann and R. Khondoker, "Security analysis of SDN WiFi applications," in *SDN and NFV Security* (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 57–71.

[27] M. Bräuning and R. Khondoker, "Analysis of SDN applications for smart grid infrastructures," in *SDN and NFV Security* (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 99–110.

[28] A. Chikhale and R. Khondoker, "Security analysis of SDN cloud applications," in *SDN and NFV Security* (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 19–38.

[29] R. Jain and R. Khondoker, "Security analysis of SDN WAN applications—B4 and IWAN," in *SDN and NFV Security* (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 111–127.

[30] C. Lee, C. Yoon, S. Shin, and S. K. Cha, "INDAGO: A new framework for detecting malicious SDN applications," in *Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP)*, Sep. 2018, pp. 220–230.

[31] C. Lee and S. Shin, "SHIELD: An automated framework for static analysis of SDN applications," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, New York, NY, USA, Mar. 2016, pp. 29–34.

[32] R. Durairajan, J. Sommers, and P. Barford, "Controller-agnostic SDN debugging," in *Proc. Conf. Emerg. Netw. Exp. Technol.*, New York, NY, USA, 2014, pp. 227–233, doi: 10.1145/2674005.2674993.

[33] Y. Li, Z. Wang, J. Yao, X. Yin, X. Shi, J. Wu, and H. Zhang, "MSAID: Automated detection of interference in multiple SDN applications," *Comput. Netw.*, vol. 153, pp. 49–62, Apr. 2019.

[34] T. Hu, P. Yi, Y. Hu, J. Lan, Z. Zhang, and Z. Li, "SAIDE: Efficient application interference detection and elimination in SDN," *Comput. Netw.*, vol. 183, Dec. 2020, Art. no. 107619.

[35] M. Latah and L. Toker, "Load and stress testing for SDN's northbound API," *Social Netw. Appl. Sci.*, vol. 2, no. 1, Dec. 2019, Art. no. 122, doi: 10.1007/s42452-019-1917-y.

[36] (2020). *GitHub-The POX Network Software Platform*. [Online]. Available: https://github.com/noxrepo/pox

[37] Community. (2017). *Ryu SDN Framework*. [Online]. Available: https://ryu-sdn.org/

[38] (2012). *Architecture—Floodlight Controller—Project Floodlight*. [Online]. Available: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview

[39] (2019). *Home—OpenDaylight*. [Online]. Available: https://www.opendaylight.org/

[40] ON Foundation. (2020). *Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions*. [Online]. Available: https://opennetworking.org/onos/%0A

[41] B. E. Ujcich, S. Jero, A. Edmundson, Q. Wang, R. Skowyra, J. Landry, A. Bates, W. H. Sanders, C. Nita-Rotaru, and H. Okhravi, "Cross-app poisoning in software-defined networking," in *Proc. ACM Conf. Comput. Commun. Secur.* New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 648–663. [Online]. Available: https://dl.acm.org/doi/10.1145/3243734.3243759

[42] R. Chang, Z. Lin, Y. Sun, and J. Xu, "MD-UCON: A multi-domain access control model for SDN northbound interfaces," *J. Phys., Conf. Ser.*, vol. 1187, no. 3. 2019, Art. no. 32091.

[43] J. Park and R. Sandhu, "Towards usage control models: Beyond traditional access control," in *Proc. 7th ACM Symp. Access Control Models Technol.*, New York, NY, USA, 2002, p. 57. [Online]. Available: http://portal.acm.org/citation.cfm?doid=507711.507722

[44] H. D. Hoang, P. T. Duy, and V.-H. Pham, "A security-enhanced monitoring system for northbound interface in SDN using blockchain," in *Proc. 10th Int. Symp. Inf. Commun. Technol.*, New York, NY, USA, 2019, pp. 197–204.

[45] Y. Tseng, M. Pattaranantakul, R. He, Z. Zhang, and F. Nait-Abdesselam, "Controller DAC: Securing SDN controller with dynamic access control," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[46] H. Padekar, Y. Park, H. Hu, and S.-Y. Chang, "Enabling dynamic access control for controller applications in software-defined networks," in *Proc. 21st ACM Symp. Access Control Models Technol.*, New York, NY, USA, Jun. 2016, pp. 51–61.

[47] B. Toshniwal, K. D. Joshi, P. Shrivastava, and K. Kataoka, "BEAM: Behavior-based access control mechanism for SDN applications," in *Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2019, pp. 1–2.

[48] Y. Tseng, F. Nait-Abdesselam, and A. Khokhar, "SENAD: Securing network application deployment in software defined networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[49] H. Cui, Z. Chen, L. Yu, K. Xie, and Z. Xia, "Authentication mechanism for network applications in SDN environments," in *Proc. 20th Int. Symp. Wireless Pers. Multimedia Commun. (WPMC)*, Dec. 2017, pp. 1–5.

[50] G. Kim, J. An, and K. Kim, "A study on authentication mechanism in SEaaS for SDN," in *Proc. 11th Int. Conf. Ubiquitous Inf. Manage. Commun.*, New York, NY, USA, Jan. 2017, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3022227.3022277

[51] C. Banse and S. Rangarajan, "A secure northbound interface for SDN applications," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2015, pp. 834–839.

[52] Y. Tseng, Z. Zhang, and F. Nait-Abdesselam, "ControllerSEPA: A security-enhancing SDN controller plug-in for OpenFlow applications," in *Parallel Distrib. Comput., Appl. Technol.*, Jul. 2016, pp. 268–273.

[53] S. B. H. Natanzi and M. R. Majma, "Secure northbound interface for SDN applications with NTRU public key infrastructure," in *Proc. IEEE 4th Int. Conf. Knowl.-Based Eng. Innov. (KBEI)*, Dec. 2017, pp. 452–458.

[54] T. Hu, Z. Zhang, P. Yi, D. Liang, Z. Li, Q. Ren, Y. Hu, and J. Lan, "SEAPP: A secure application management framework based on REST API access control in SDN-enabled cloud environment," *J. Parallel Distrib. Comput.*, vol. 147, pp. 108–123, Jan. 2021.

[55] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017.

[56] L. Zhu, M. Monjurul Karim, K. Sharif, F. Li, X. Du, and M. Guizani, "SDN controllers: Benchmarking performance evaluation," 2019, *arXiv:1902.04491*. [Online]. Available: http://arxiv.org/abs/1902.04491

[57] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting security attacks in software-defined networks," in *Proc. NDSS*, vol. 15, 2015, pp. 8–11.

[58] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *J. Netw. Comput. Appl.*, vol. 103, pp. 101–118, Feb. 2018.

[59] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for openflow," in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, vol. 3, 2010, pp. 1–8.

[60] N. Katta, H. Zhang, M. Freedman, and J. Rexford, "Ravana: Controller fault-tolerance in software-defined networking," in *Proc. Symp. Softw. Defined Netw. (SDN)*, New York, NY, USA, Jun. 2015, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2774993.2774996

[61] A. D. Ferguson, A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi, "Participatory networking: An API for application control of SDNs," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 327–338, Oct. 2013.

[62] P. Hunt, M. Konar, Y. Grid, F. P. Junqueira, B. Reed, and Y. Research, "ZooKeeper: Wait-free coordination for internet-scale systems," in *Proc. USENIX Annu. Tech. Conf.*, Jun. 2010, p. 9.

[63] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. Annu. Tech. Conf.*, 2014, pp. 305–319.

[64] R. Macedo, R. de Castro, A. Santos, Y. Ghamri-Doudane, and M. Nogueira, "Self-organized SDN controller cluster conformations against DDoS attacks effects," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[65] Z. Latif, K. Sharif, F. Li, M. Monjurul Karim, and Y. Wang, "A comprehensive survey of interface protocols for software defined networks," 2019, *arXiv:1902.07913*. [Online]. Available: http://arxiv.org/abs/1902.07913

[66] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, "SDNI: A message exchange protocol for software defined networks (SDNS) across multiple domains," in *Proc. IETF Draft, Work Progress*, 2012, pp. 1–5.

[67] (2021). *NVD-CVE-2018-1132*. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-1132

[68] P. Lin, J. Bi, Z. Chen, Y. Wang, H. Hu, and A. Xu, "WE-bridge: West-east bridge for SDN inter-domain network peering," in *Proc. Inst. Elect. Electron. Eng.*, 2014, pp. 111–112.

[69] K. Phemius, M. Bouet, and J. Leguay, "DISCO: Distributed multi-domain SDN controllers," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp., Manage. Softw. Defined World*, 2014, pp. 1–4.

[70] F. Benamrane, M. B. Mamoun, and R. Benaini, "New method for controller-to-controller communication in distributed SDN architecture," *Int. J. Commun. Netw. Distrib. Syst.*, vol. 19, no. 3, pp. 357–367, 2017.

[71] Z. Allybokus, K. Avrachenkov, J. Leguay, and L. Maggi, "Multi-path alpha-fair resource allocation at scale in distributed software-defined networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2655–2666, Dec. 2018.

[72] H. Yu, H. Qi, and K. Li, "WECAN: An Efficient west-east control associated network for large-scale SDN systems," *Mobile Netw. Appl.*, vol. 25, no. 1, pp. 114–124, Feb. 2020, doi: 10.1007/s11036-018-1194-9.

[73] F. Benamrane, M. Ben Mamoun, and R. Benaini, "An east-west interface for distributed SDN control plane: Implementation and evaluation," *Comput. Electr. Eng.*, vol. 57, pp. 162–175, Jan. 2017.

[74] J. H. Lam, S. G. Lee, H. J. Lee, and Y. E. Oktian, "Securing distributed SDN with IBC," in *Proc. Int. Conf. Ubiquitous Future Netw.*, Aug. 2015, Aug. 2015, pp. 921–925.

[75] S. B. Hashemi Natanzi and M. R. Majma, "Secure distributed controllers in SDN based on ECC public key infrastructure," in *Proc. Int. Conf. Electr. Comput. Technol. Appl. (ICECTA)*, Nov. 2017, pp. 1–5.

[76] R. K. Arbettu, R. Khondoker, K. Bayarou, and F. Weber, "Security analysis of OpenDaylight, ONOS, rosemary and ryu SDN controllers," in *Proc. 17th Int. Telecommun. Netw. Strategy Planning Symp. (Networks)*, Sep. 2016, pp. 37–44.

[77] S. Scott-Hayward, "Trailing the snail: SDN controller security evolution," 2017, *arXiv:1711.08406*. [Online]. Available: http://arxiv.org/abs/1711.08406

[78] (2021). *Security-Mode ONOS—ONOS—Wiki*. [Online]. Available: https://wiki.onosproject.org/display/ONOS/Security-Mode+ONOS

[79] V. H. Dixit, A. Doupé, Y. Shoshitaishvili, Z. Zhao, and G.-J. Ahn, "AIM-SDN: Attacking information mismanagement in SDN-datastores," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 664–676.

[80] S. Secci, A. Diamanti, J. M. V. Sanchez, M. T. Bah, P. Vizarreta, C. M. Machuca, S. Scott-Hayward, and D. Smith, "Security and performance comparison of ONOS and ODL controllers," Open Netw. Found., Tech. Rep., 2019.

[81] *NVD-CVE-2018-1078*. Accessed: Jun. 1, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-1078

[82] *CVE-CVE-2017-1000411*. Accessed: Apr. 22, 2021. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2017-1000411

[83] A. Bidaj, T. Aura, and L. Røstad, "Security testing SDN controllers," School Sci., Aalto Univ., Espoo, Finland, Tech. Rep., Jun. 2016.

[84] *NVD-CVE-2015-1610*. Accessed: Dec. 14, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/cve-2015-1610

[85] *NVD-CVE-2015-1611*. Accessed: Dec. 14, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2015-1611

[86] *CVE-2014-5035-Opendaylight*. Accessed: Jun. 11, 2020. [Online]. Available: https://seclists.org/fulldisclosure/2014/Aug/34

[87] OWASP. (2020). *XML External Entity (XXE) Processing|OWASP*. Accessed: Jul. 10, 2020. [Online]. Available: https://owasp.org/www-community/vulnerabilities/XML_External_Entity_XXE_Processing

[88] *OSS-SEC: OpenDayLight: Password Change Doesn't Result in Karaf Clearing Cache, Allowing Old Password to Still be Used (CVE-2017-1000406*. Accessed: Dec. 14, 2020. [Online]. Available:https://seclists.org/oss-sec/2017/q4/320

[89] *NVD-CVE-2014-8149*. Accessed: Apr. 5, 2021. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2014-8149

[90] *Security Considerations—OpenDaylight Documentation Aluminum documentation*. Accessed: Feb. 10, 2021. [Online]. Available: https://docs.opendaylight.org/en/stable-aluminium/getting-started-guide/security_considerations.html#overview-of-opendaylight-security

[91] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–9.

[92] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 566–578, Mar. 2019.

[93] J. Malik, A. Akhunzada, I. Bibi, M. Imran, A. Musaddiq, and S. W. Kim, "Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in SDN," *IEEE Access*, vol. 8, pp. 134695–134706, 2020.

[94] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, and M. Ghogho, "Intrusion detection in sdn-based networks: Deep recurrent neural network approach," in *Deep Learning Applications for Cyber Security* (Advanced Sciences and Technologies for Security Applications). Cham, Switzerland: Springer, 2019, pp. 175–195. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-13057-2_8

[95] *NVD—CVE-2017-13763*. Accessed: Dec. 14, 2020. [Online]. Available:https://nvd.nist.gov/vuln/detail/CVE-2017-13763

[96] *NVD—CVE-2018-1000615*. Accessed: Dec. 14, 2020. [Online]. Available:https://nvd.nist.gov/vuln/detail/CVE-2018-1000615

[97] *NVD—CVE-2015-7516*. Accessed: Dec. 14, 2020. [Online]. Available:https://nvd.nist.gov/vuln/detail/CVE-2015-7516

[98] *NVD-CVE-2018-1000616*. Accessed: Dec. 14, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-1000616

[99] *NVD-CVE-2018-1000614*. Accessed: Dec. 14, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-1000614

[100] *NVD-CVE-2017-1000081*. Accessed: Dec. 15, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-1000081

[101] *NVD-CVE-2017-1000080*. Accessed: Dec. 15, 2020. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-1000080

[102] *NVD-CVE-2019-13624*. Accessed: Apr. 23, 2021. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-13624

[103] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019. [Online]. Available: https://link.springer.com/articles/10.1186/s42400-019-0038-7.

[104] (2018). *OpenFlow Conformance Certification*. [Online]. Available: https://www.opennetworking.org/product-certification/

[105] *RFC 7047-The Open vSwitch Database Management Protocol*. Accessed: Feb. 9, 2021. [Online]. Available: https://tools.ietf.org/html/rfc7047

[106] (2014). *OpFlex: An Open Source Approach*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-731304.html

[107] *RFC 6241—Network Configuration Protocol (NETCONF)*. [Online]. Available: https://tools.ietf.org/html/rfc6241

[108] *RFC 7391—Forwarding and Control Element Separation (ForCES) Protocol Extensions*. [Online]. Available: https://tools.ietf.org/html/rfc7391

[109] (2015). *OpenFlow Switch Specification Version 1.5.1*. [Online]. Available: http://www.opennetworking.org

[110] (2018). *CVE—CVE-2020-1968*. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-1968

[111] *CVE—CVE-2014-8730*. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-8730

[112] B. Agborubere and E. Sanchez-Velazquez, "OpenFlow communications and TLS security in software-defined networks," in *Proc. IEEE Int. Conf. Internet Things*, Jan. 2018, pp. 560–566.

[113] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, New York, NY, USA, Aug. 2013, pp. 151–152. [Online]. Available: http://www.wireshark.org/security/

[114] J. Lam, S.-G. Lee, H.-J. Lee, and Y. E. Oktian, "Securing SDN southbound and data plane communication with IBC," *Mobile Inf. Syst.*, vol. 2016, Aug. 2016, Art. no. 1708970.

[115] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Rules placement problem in OpenFlow networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1273–1286, 2nd Quart., 2016.

[116] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, "Toward adaptive and scalable openflow-SDN flow control: A survey," *IEEE Access*, vol. 7, pp. 107346–107379, 2019.

[117] E. Marin, N. Bucciol, and M. Conti, "An in-depth look into SDN topology discovery mechanisms: Novel attacks and practical countermeasures," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, Nov. 2019, pp. 1101–1114.

[118] P. Congdon, P. Networking, P. Blatherwick, and M. Networks. (2004). *802.1AB Overview Link Layer Discovery Protocol*. [Online]. Available: http://www.ieee802.org/3/frame_study/0409/blatherwick_1_0409.pdf

[119] L. O. Aday, C. Pastor. (2015). *Current Trends of Topology Discovery in OpenFlow-based Software Defined Networks*. [Online]. Available: http://upcommons.upc.edu/handle/2117/77672

[120] Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, and C. Yang, "Traffic engineering in software-defined networking: Measurement and management," *IEEE Access*, vol. 4, pp. 3246–3256, 2016.

[121] T.-H. Nguyen and M. Yoo, "Attacks on host tracker in SDN controller: Investigation and prevention," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2016, pp. 610–612.

[122] T.-H. Nguyen and M. Yoo, "Analysis of link discovery service attacks in SDN controller," in *Proc. IEEE ICOIN*, Jan. 2017, pp. 259–261.

[123] A. Azzouni, R. Boutaba, N. T. M. Trang, G. Pujolle, S. Deng, X. X. Gao, Z. Lu, and X. X. and Gao, "sOFTDP: Secure and efficient topology discovery protocol for SDN," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 695–705, May 2017. [Online]. Available: http://arxiv.org/abs/1705.04527

[124] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in *Proc. NDSS*, vol. 15, 2015, pp. 8–11.

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

[126] R. Skowyra, L. Xu, G. Gu, V. Dedhia, T. Hobson, H. Okhravi, and J. Landry, "Effective topology tampering attacks and defenses in software-defined networks," in *Proc. 48th IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jul. 2018, pp. 374–385.

[127] P. Shrivastava, A. Agarwal, and K. Kataoka, "Detection of topology poisoning by silent relay attacker in SDN," in *Proc. 24th Annu. Int. Conf. Mobile Comput. Netw.*, New York, NY, USA, Oct. 2018, pp. 792–794, doi: 10.1145/3241539.3267763.

[128] T. Wang and H. Chen, "SGuard: A lightweight SDN safe-guard architecture for DoS attacks," *China Commun.*, vol. 14, no. 6, pp. 113–125, 2017.

[129] T. Y. Lin, J. P. Wu, P. H. Hung, C. H. Shao, Y. T. Wang, Y. Z. Cai, and M. H. Tsai, "Mitigating SYN flooding attack and ARP spoofing in SDN data plane," in *Proc. 21st Asia–Pacific Netw. Oper. Manage. Symp.*, Sep. 2020, pp. 114–119.

[130] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the vulnerability of flow table overflow in software-defined network: Attack model, evaluation, and defense," *Secur. Commun. Netw.*, vol. 2018, Jan. 2018, Art. no. 4760632.

[131] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in software defined networks," *Comput. Netw.*, vol. 71, pp. 1–30, Jun. 2014.

[132] A. Capone, C. Cascone, A. Q. T. Nguyen, B. Sansò, and P. Di Milano, "Detour planning for fast and reliable fault recovery in SDN," Dept. Génie, Open-State Polytechnique Montréal, Tech. Rep. DRCN'15, Mar. 25, 2015.

[133] X. Zhang, L. Cui, K. Wei, F. P. Tso, Y. Ji, and W. Jia, "A survey on stateful data plane in software defined networks," *Comput. Netw.*, vol. 184, Jan. 2021, Art. no. 107597.

[134] S. Xiang, H. Zhu, L. Xiao, and W. Xie, "Modeling and verifying TopoGuard in OpenFlow-based software defined networks," in *Proc. Int. Symp. Theor. Aspects Softw. Eng. (TASE)*, Aug. 2018, pp. 84–91.

[135] A. S. Alshra'a and J. Seitz, "Using INSPECTOR device to stop packet injection attack in SDN," *IEEE Commun. Lett.*, vol. 23, no. 7, pp. 1174–1177, Jul. 2019.

[136] M. Imran, M. H. Durad, F. A. Khan, and H. Abbas, "DAISY: A detection and mitigation system against denial-of-service attacks in software-defined networks," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1933–1944, Jun. 2020.

[137] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS attack prevention extension in software-defined networks," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2015, pp. 239–250.

[138] X. Huang, X. Du, and B. Song, "An effective DDoS defense scheme for SDN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[139] M. Zhang, G. Li, L. Xu, J. Bi, G. Gu, and J. Bai, "Control plane reflection attacks in SDNs: New attacks and countermeasures," in *Research in Attacks, Intrusions, and Defenses* (Lecture Notes in Computer Science), vol. 11050. Cham, Switzerland: Springer-Verlag, Sep. 2018, pp. 161–183, doi: 10.1007/978-3-030-00470-5_8.

[140] J. Xu, L. Wang, and Z. Xu, "An enhanced saturation attack and its mitigation mechanism in software-defined networking," *Comput. Netw.*, vol. 169, Mar. 2020, Art. no. 107092.

[141] T. Wang, H. Chen, G. Cheng, and Y. Lu, "SDNManager: A safeguard architecture for SDN DoS attacks based on bandwidth prediction," *Secur. Commun. Netw.*, vol. 2018, pp. 1–16, Oct. 2018.

[142] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, and D. and Burgos, "An evolutionary SVM model for DDOS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 132502–132513, 2020.

[143] G. S. Aujla, M. Singh, A. Bose, N. Kumar, G. Han, and R. Buyya, "BlockSDN: Blockchain-as-a-service for software defined networking in smart city applications," *IEEE Netw.*, vol. 34, no. 2, pp. 83–91, Mar. 2020.

[144] S.-C. Su, Y.-R. Chen, S.-C. Tsai, and Y.-B. Lin, "Detecting P2P botnet in software defined networks," *Secur. Commun. Netw.*, vol. 2018, Jan. 2018, Art. no. 4723862.

[145] A. N. Viet, L. P. Van, H.-A.-N. Minh, H. D. Xuan, N. P. Ngoc, and T. N. Huu, "Mitigating HTTP GET flooding attacks in SDN using NetFPGA-based OpenFlow switch," in *Proc. 14th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, Jun. 2017, pp. 660–663.

[146] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2015, pp. 77–81.

[147] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "SAFETY: Early detection and mitigation of TCP SYN flood utilizing entropy in SDN," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1545–1559, Dec. 2018.

[148] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. Rodrigues, B. Sahoo, and R. Dash, "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," *Future Gener. Comput. Syst.*, vol. 89, pp. 685–697, Dec. 2018.

[149] D. Wu, J. Li, S. K. Das, J. Wu, Y. Ji, and Z. Li, "A novel distributed denial-of-service attack detection scheme for software defined networking environments," in *IEEE Int. Conf. Commun.*, Jul. 2018, pp. 1–6.

[150] R. B. Shohani and S. A. Mostafavi, "Introducing a new linear regression based method for early DDoS attack detection in SDN," in *Proc. 6th Int. Conf. Web Res. (ICWR)*, Apr. 2020, pp. 126–132.

[151] S. Badotra and S. N. Panda, "SNORT based early DDoS detection system using opendaylight and open networking operating system in software defined networking," *Cluster Comput.*, vol. 24, no. 1, pp. 501–513, Mar. 2021, doi: 10.1007/s10586-020-03133-y.

[152] L. Barki, A. Shidling, N. Meti, D. G. Narayan, and M. M. Mulla, "Detection of distributed denial of service attacks in software defined networks," in *Proc. Int. Conf. Adv. Comput., Commun. Informat.*, Nov. 2016, pp. 2576–2581.

[153] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, "Detection and defense of DDoS attack-based on deep learning in OpenFlow-based SDN," *Int. J. Commun. Syst.*, vol. 31, no. 5, p. e3497, Mar. 2018.

[154] A. Banitalebi Dehkordi, M. Soltanaghaei, and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SDN," *J. Supercomput.*, vol. 77, no. 3, pp. 2383–2415, Mar. 2021, doi: 10.1007/s11227-020-03323-w.

[155] R. Mohammadi, R. Javidan, M. Keshtgary, M. Conti, and C. Lal, "Practical extensions to countermeasure DoS attacks in software defined networking," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2017, pp. 1–6.

[156] S. Khorsandroo and A. S. Tosun, "Time inference attacks on software defined networks: Challenges and countermeasures," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 342–349.

[157] A. R. Chavez, W. M. S. Stout, and S. Peisert, "Techniques for the dynamic randomization of network attributes," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Sep. 2015, pp. 1–6.

[158] B. G. Assefa and Ö. Özkasap, "A survey of energy efficiency in SDN: Software-based methods and optimization models," *J. Netw. Comput. Appl.*, vol. 137, pp. 127–143, Jul. 2019.

[159] M. Karakus and A. Durresi, "Quality of service (QoS) in software defined networking (SDN): A survey," *J. Netw. Comput. Appl.*, vol. 80, pp. 200–218, Feb. 2017.

[160] S. Xie, C. Xing, G. Zhang, and J. Zhao, "A table overflow LDoS attack defending mechanism in software-defined networks," *Secur. Commun. Netw.*, vol. 2021, pp. 1–16, Jan. 2021.

[161] F. Nife and Z. Kotulski, "New SDN-oriented authentication and access control mechanism," in *Computer Networks*, vol. 860. Cham, Switzerland: Springer-Verlag, 2018, pp. 74–88.

[162] K. Benzekki, A. El Fergougui, and A. El Belrhiti El Alaoui, "Devolving IEEE 802.1X authentication capability to data plane in software-defined networking (SDN) architecture," *Secur. Commun. Netw.*, vol. 9, no. 17, pp. 4369–4377, Nov. 2016. [Online]. Available: http://doi.wiley.com/10.1002/sec.1613

[163] J. Matias, J. Garay, A. Mendiola, N. Toledo, and E. Jacob, "FlowNAC: Flow-based network access control," in *Proc. 3rd Eur. Workshop Softw. Defined Netw. (EWSDN)*, Sep. 2014, pp. 79–84.

[164] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, and K.-K.-R. Choo, "P4-to-blockchain: A secure blockchain-enabled packet parser for software defined networking," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101629.

[165] R.-H. Hwang, V.-L. Nguyen, and P.-C. Lin, "StateFit: A security framework for SDN programmable data plane model," in *Proc. 15th Int. Symp. Pervas. Syst., Algorithms Netw. (I-SPAN)*, Oct. 2018, pp. 168–173.

[166] B. Lewis, M. Broadbent, and N. Race, "P4ID: P4 enhanced intrusion detection," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2019, pp. 1–4.

[167] A. D. S. Ilha, A. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Euclid: A fully in-network, P4-based approach for real-time DDoS attack detection and mitigation," *IEEE Trans. Netw. Service Manage.*, vol. 4537, pp. 1–19, Dec. 2020.

[168] F. Hauser, M. Schmidt, M. Haberle, and M. Menth, "P4-MACsec: Dynamic topology monitoring and data layer protection with MACsec in P4-based SDN," *IEEE Access*, vol. 8, pp. 58845–58858, 2020.

[169] J. Xing, A. Chen, and T. S. Eugene Ng, "Secure state migration in the data plane," in *Proc. ACM SIGCOMM Workshop Secure Program. Netw. Infrastruct.*, Aug. 2020, pp. 28–34, doi: 10.1145/3405669.3405822.

[170] J. Xing, W. Wu, and A. Chen, "Architecting programmable data plane defenses into the network with fastFlex," in *Proc. 18th ACM Workshop Hot Topics Netw.*, New York, NY, USA, 2019, pp. 161–169, doi: 10.1145/3365609.3365860.

[171] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted DDoS attack detection with P4 language," in *Proc. ICC - IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[172] C. Sun, J. Bi, H. Chen, H. Hu, Z. Zheng, S. Zhu, and C. Wu, "SDPA: Toward a stateful data plane in software-defined networking," *IEEE/ACM Trans. Netw.*, vol. 25, no. 6, pp. 3294–3308, Dec. 2017.

[173] M. T. Arashloo, Y. Koral, M. Greenberg, J. Rexford, and D. Walker, "SNAP: Stateful network-wide abstractions for packet processing," in *Proc. SIGCOMM*, Florianopolis, Brazil, Aug. 2016, pp. 29–43.

[174] M. Moshref, A. Bhargava, A. Gupta, M. Yu, and R. Govindan, "Flow-level state transition as a new switch primitive for SDN," in *Proc. 3rd workshop Hot topics Softw. defined Netw.*, New York, NY, USA, Aug. 2014, doi: 10.1145/2620728.2620729.

[175] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "Openstate: Programming platform-independent stateful openflow applications inside the switch," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 44–51, Apr. 2014. [Online]. Available: https://dl.acm.org/doi/10.1145/2602204.2602211

[176] G. Bianchi, M. Bonola, S. Pontarelli, D. Sanvito, A. Capone, and C. Cascone, "Open packet processor: A programmable architecture for wire speed platform-independent stateful in-network processing," 2016, *arXiv:1605.01977*. [Online]. Available: http://arxiv.org/abs/1605.01977

[177] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–95, Jul. 2014. [Online]. Available: https://dl.acm.org/doi/10.1145/2656877.2656890

[178] P. Krongbaramee and Y. Somchit, "Implementation of SDN stateful firewall on data plane using open vSwitch," in *Proc. 15th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE)*, Jul. 2018, pp. 1–5.

[179] M. Caprolu, S. Raponi, and R. Di Pietro, "FORTRESS: An efficient and distributed firewall for stateful data plane SDN," *Secur. Commun. Netw.*, vol. 2019, Oct. 2019, Art. no. 6874592. [Online]. Available: https://doi.org/10.1155/2019/6874592

[180] C. Cascone, L. Pollini, D. Sanvito, and A. Capone, "Traffic management applications for stateful SDN data plane," in *Proc. 4th Eur. Workshop Softw. Defined Netw.*, Sep. 2015, pp. 85–90.

[181] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable load balancing using programmable data planes," in *Proc. Symp. Softw. Defined Netw. (SDN) Res.*, New York, NY, USA, Mar. 2016, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2890955.2890968

[182] C. Hernandez Benet, A. J. Kassler, T. Benson, and G. Pongracz, "MP-HULA: Multipath transport aware load balancing using programmable data planes," in *Proc. Morning Workshop Netw. Comput.*, vol. 18. New York, NY, USA: ACM Press, 2018, pp. 7–13, doi: 10.1145/3229591.3229596.

[183] M. Bonola, R. Bifulco, L. Petrucci, S. Pontarelli, A. Tulumello, and G. Bianchi, "Implementing advanced network functions for datacenters with stateful programmable data planes," in *Proc. IEEE Workshop Local Metrop. Area Netw.*, Jul. 2017, pp. 1–6.

[184] Z. Zaidi, V. Friderikos, Z. Yousaf, S. Fletcher, M. Dohler, and H. Aghvami, "Will SDN be part of 5G?" *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3220–3258, 1st Quart., s2018.

[185] *Hp News-HP Launches Industry's First SDN App Store, Unleashing New Wave of Networking Innovations*. Accessed: Mar. 2, 2021. [Online]. Available: https://www8.hp.com/us/en/hp-news/press-release.html?id=1798074#.yd6ag2hki%uk

[186] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 333–354, 1st Quart., 2018.

[187] T. Zhang, A. Bianco, S. De Domenico, and P. Giaccone, "The role of inter-controller traffic for placement of distributed SDN controllers," *Comput. Commun.*, vol. 113, pp. 1–13, May 2016. [Online]. Available: http://arxiv.org/abs/1605.09268

[188] R. Hanmer, L. Jagadeesan, V. Mendiratta, and H. Zhang, "Friend or foe: Strong consistency vs. Overload in high-availability distributed systems and SDN," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2018, pp. 59–64.

[189] M. Roohitavaf, K. Ren, J. S. Ahn, G. Zhang, S. S. Kulkarni, W. H. Kang, and S. Ben-Romdhane, "Session guarantees with raft and hybrid logical clocks," in *ACM Int. Conf. Proc. Ser.*, New York, NY, USA, Jan. 2019, pp. 100–109. [Online]. Available: https://dl.acm.org/doi/10.1145/3288599.3288619

[190] E. Sakic and W. Kellerer, "Response time and availability study of RAFT consensus in distributed SDN control plane," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 1, pp. 304–318, Feb. 2019. [Online]. Available: http://arxiv.org/abs/1902.02537.

[191] R. Hanmer, S. Liu, L. Jagadeesan, and M. R. Rahman, "Death by babble: Security and fault tolerance of distributed consensus in high-availability softwarized networks," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, Jun. 2019, pp. 266–270.

[192] C. Zhang, P. Sun, G. Hu, and L. Zhu, "RETCAM: An efficient TCAM compression model for flow table of OpenFlow," *J. Commun. Netw.*, vol. 22, no. 6, pp. 484–492, Dec. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9321162/

[193] M. Rifai, N. Huin, C. Caillouet, F. Giroire, J. Moulierac, D. L. Pacheco, and G. Urvoy-Keller, "Minnie: An SDN world with few compressed forwarding rules," *Comput. Netw.*, vol. 121, pp. 185–207, Jul. 2017.

[194] M. Yu, T. He, P. McDaniel, and Q. K. Burke, "Flow table security in SDN: Adversarial reconnaissance and intelligent attacks," in *Proc. IEEE Conf. Comput. Commun.*, Jul. 2020, pp. 1519–1528.

[195] D. Scholz, S. Gallenmüller, H. Stubbe, and G. Carle, "SYN flood defense in programmable data planes," in *Proc. 3rd P4 Workshop Eur.*, New York, NY, USA, Dec. 2020, pp. 13–20.

[196] (2021). *5G PPP Technology Board Edge Computing for 5G Networks Edge Computing for 5G Networks*. [Online]. Available: https://doi.org/10.5281/zenodo.3698117

[197] S. Lee, S. Woo, J. Kim, V. Yegneswaran, P. Porras, and S. Shin, "AudiSDN: Automated detection of network policy inconsistencies in software-defined networks," in *Proc. INFOCOM*, Jul. 2020, pp. 1788–1797.

[198] S. Lee, C. Yoon, C. Lee, S. Shin, V. Yegneswaran, and P. Porras, "DELTA: A security assessment framework for software-defined networks," in *Proc. NSDD*, 2017, pp. 1–5, doi: 10.14722/ndss.2017.23457.

[199] S. Lee, J. Kim, S. Shin, P. Porras, and V. Yegneswaran, "Athena: A framework for scalable anomaly detection in software-defined networks," in *Proc. 47th IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2017, pp. 249–260.

[200] L. Fawcett, S. Scott-Hayward, M. Broadbent, A. Wright, and N. Race, "Tennison: A distributed SDN framework for scalable network security," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 12, pp. 2805–2818, Dec. 2018.

[201] K. K. Karmakar, V. Varadharajan, and U. Tupakula, "On the design and implementation of a security architecture for software defined networks," in *Proc. IEEE 18th Int. Conf. High Perform. Comput. Commun.*, Dec. 2016, pp. 671–678.

[202] H. Krawczyk, K. G. Paterson, and H. Wee, "On the security of the TLS protocol: A systematic analysis," in *Advances in Cryptology* (Lecture Notes in Computer Science), vol. 8042. Berlin, Germany: Springer, 2013, pp. 429–448.

[203] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 393–430, 1st Quart., 2019.

[204] R. Durner and W. Kellerer. (2021). *The Cost of Security in the SDN Control Plane*. [Online]. Available: http://dx.doi.org/10.1145/2842665.2843563

[205] H. Tahaei, R. B. Salleh, M. F. Ab Razak, K. Ko, and N. B. Anuar, "Cost effective network flow measurement for software defined networks: A distributed controller scenario," *IEEE Access*, vol. 6, pp. 5182–5198, 2018.

[206] W. Feng, C. Liu, B. Cheng, and J. Chen, "Secure and cost-effective controller deployment in multi-domain SDN with baguette," *J. Netw. Comput. Appl.*, vol. 178, Mar. 2021, Art. no. 102969.

[207] P. C. da Rocha Fonseca and E. S. Mota, "A survey on fault management in software-defined networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2284–2321, 4th Quart., 2017.

[208] Y. Yu, X. Li, X. Leng, L. Song, K. Bu, Y. Chen, J. Yang, L. Zhang, K. Cheng, and X. Xiao, "Fault management in software-defined networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 349–392, 1st Quart., 2019.

[209] *ONOS Security and Performance Analysis*. Accessed: Jan. 6, 2021. [Online]. Available: https://opennetworking.org/wp-content/uploads/2017/07/ONOS-security-and-performance-analysis-brigade-report-no1.pdf

[210] B. Yigit, G. Gur, B. Tellenbach, and F. Alagoz, "Secured communication channels in software-defined networks," *IEEE Commun. Mag.*, vol. 57, no. 10, pp. 63–69, Oct. 2019.

[211] OWASP. (2020). *OWASP Top Ten Web Application Security Risks|OWASP*. [Online]. Available: https://owasp.org/www-project-top-ten/

[212] T. K. Dasaklis, F. Casino, and C. Patsakis, "SoK: Blockchain solutions for forensics," in *Technology Development for Security Practitioners*. Cham, Switzerland: Springer, 2021, pp. 21–40.

[213] P. T. Duy, H. Do Hoang, D. T. Thu Hien, N. Ba Khanh, and V.-H. Pham, "SDNLog-foren: Ensuring the integrity and tamper resistance of log files for SDN forensics using blockchain," in *Proc. 6th NAFOSTED Conf. Inf. Comput. Sci. (NICS)*, Dec. 2019, pp. 416–421.

[214] M. Lagrasse, A. Singh, H. Munkhondya, A. Ikuesan, and H. Venter, "Digital forensic readiness framework for software-defined networks using a trigger-based collection mechanism," in *Proc. 15th Int. Conf. Cyber Warfare Secur.*, 2020, pp. 296–305.

[215] (1990). *The Deming Management Method*. [Online]. Available: https://books.google.es/books/about/The_Deming_Management_Method.html?id=4tPlxq76ssYC&redir_esc=y

[216] A. Gillies, "Improving the quality of information security management systems with ISO27000," *TQM J.*, vol. 23, no. 4, pp. 367–376, Jun. 2011.

[217] I. Farris, T. Taleb, Y. Khettab, and J. Song, "A survey on emerging SDN and NFV security mechanisms for IoT systems," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 812–837, 1st Quart., 2019.

[218] J. L. Luo, S. Z. Yu, and S. J. Peng, "SDN/NFV-based security service function tree for cloud," *IEEE Access*, vol. 8, pp. 38538–38545, 2020.

[219] K. S. Sahoo and D. Puthal, "SDN-assisted DDoS defense framework for the internet of multimedia things," *ACM Trans. Multimedia Comput., Commun. Appl.*, vol. 16, no. 3s, pp. 1–18, Jan. 2021, doi: 10.1145/3394956.

[220] P. K. Sharma, S. Singh, Y. S. Jeong, and J. H. Park, "DistBlockNet: A distributed blockchains-based secure SDN architecture for IoT networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, Sep. 2017.

[221] I. H. Abdulqadder, S. Zhou, D. Zou, I. T. Aziz, and S. M. A. Akber, "Bloc-sec: Blockchain-based lightweight security architecture for 5G/B5G enabled SDN/NFV cloud of IoT," in *Proc. IEEE 20th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2020, pp. 499–507.

[222] A. Rahman, M. K. Nasir, Z. Rahman, A. Mosavi, S. Shahab, and B. Minaei-Bidgoli, ''Distblock building: A distributed blockchain-based SDN-IoT network for smart building management,'' *IEEE Access*, vol. 8, pp. 140008–140018, 2020.
[223] S. R. Basnet and S. Shakya, ''BSS: Blockchain security over software defined network,'' in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, May 2017, pp. 720–725.

MARÍA B. JIMÉNEZ received the degree in systems engineering (telematics) from Universidad Politécnica Salesiana, Ecuador, in 2010, and the M.B.A. degree, in 2017. She is currently pursuing the Ph.D. degree. Since 2007, she has been participated in multiple telematics engineering implementation projects in hydrocarbon, aeronautical, and social areas. Her research interests include software-defined network security and network functions virtualization.

DAVID FERNÁNDEZ received the M.S. degree in telecommunications engineering and the Ph.D. degree in telematics engineering from the Universidad Politécnica de Madrid (UPM), Spain, in 1988 and 1993, respectively. Since 1995, he has been an Associate Professor with the Department of Telematics Systems Engineering (DIT), UPM. His current research interests include software-defined networks, network virtualization, cloud computing datacenter technologies, and network security.

JORGE EDUARDO RIVADENEIRA (Member, IEEE) received the degree in electronic and networking engineering from the National Polytechnic School, Ecuador, in 2013, and the M.Sc. degree (Hons.) in cyber security from the University of Southampton, U.K., in 2016. He is currently pursuing the Ph.D. degree with the Department of Informatics Engineering, University of Coimbra, Portugal. He is currently a Researcher with the Centre for Informatics and Systems of the University of Coimbra (CISUC). His main research interests include the Internet of Things (IoT), machine-to-machine, privacy-aware systems, security, and cryptography.

LUIS BELLIDO received the M.S. and Ph.D. degrees in telecommunications engineering from the Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 1994 and 2004, respectively. He is currently an Associate Professor with UPM, specializing in the fields of computer networking, internet technologies, and quality of service. His current research interests include multimedia applications, mobile networks, software-defined networks, and virtualization.

ANDRÉS CÁRDENAS (Member, IEEE) received the master's degree in network engineering and telematic services from the Universidad Politécnica de Madrid, Spain, where he is currently pursuing the Ph.D. degree with the Departamento de Ingeniería de Servicios Telemáticos (DIT). His current research interests include network slicing, network virtualization, software-defined networking, cloud computing, and 5G mobile networks.