Journal of Network and Computer Applications 61 (2016) 199–221

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Review

Secure and dependable software defined networks

Adnan Akhunzada [a,*], Abdullah Gani [a], Nor Badrul Anuar [a], Ahmed Abdelaziz [a], Muhammad Khurram Khan [b], Amir Hayat [c], Samee U. Khan [d]

[a] Centre for Mobile Cloud Computing Research (C4MCCR), Faculty of Computer Science and Information Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
[b] Center of Excellence in Information Assurance (CoEIA), King Saud University, Saudi Arabia
[c] Applied Security Engineering Research Group, Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan
[d] Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58108-6050, USA

A R T I C L E I N F O

Article history:
Received 17 June 2015
Received in revised form
26 November 2015
Accepted 26 November 2015
Available online 4 December 2015

Keywords:
Software defined networks
Programmable networks
Open Flow
Policy enforcement
Middle-boxes

A B S T R A C T

The revolutionary concept of Software Defined Networks (SDNs) potentially provides flexible and well-managed next-generation networks. All the hype surrounding the SDNs is predominantly because of its centralized management functionality, the separation of the control plane from the data forwarding plane, and enabling innovation through network programmability. Despite the promising architecture of SDNs, security was not considered as part of the initial design. Moreover, security concerns are potentially augmented considering the logical centralization of network intelligence. Furthermore, the security and dependability of the SDN has largely been a neglected topic and remains an open issue. The paper presents a broad overview of the security implications of each SDN layer/interface. This paper contributes further by devising a contemporary layered/interface taxonomy of the reported security vulnerabilities, attacks, and challenges of SDN. We also highlight and analyze the possible threats on each layer/interface of SDN to help design secure SDNs. Moreover, the ensuing paper contributes by presenting the state-of-the-art SDNs security solutions. The categorization of solutions is followed by a critical analysis and discussion to devise a comprehensive thematic taxonomy. We advocate the production of secure and dependable SDNs by presenting potential requirements and key enablers. Finally, in an effort to anticipate secure and dependable SDNs, we present the ongoing open security issues, challenges and future research directions.

Contents

* Corresponding author. Tel.: +60 1116431032; fax: +60 379579249.
E-mail addresses: a.adnan@siswa.um.edu.my (A. Akhunzada), abdullahgani@ieee.org (A. Gani), badrul@um.edu.my (N.B. Anuar), ahmedaziz@siswa.um.edu.my (A. Abdelaziz), mkhurram@ksu.edu.sa (M.K. Khan), amir.hayat@comsats.edu.pk (A. Hayat), samee.khan@ndsu.edu (S.U. Khan).

http://dx.doi.org/10.1016/j.jnca.2015.11.012
1084-8045/© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

The emergence of the software-defined network (SDN) paradigm simplifies network management and enables innovation through network programmability. SDNs have given rise to radical changes in the traditional vertical integration model of a network by decoupling the forwarding hardware (data plane) from the control logic of the network (control plane) (Kreutz et al., 2015; Fei et al., 2014; Xia et al., 2014; Liu et al., 2015). The data plane, which consists of switches and routers, is responsible only for forwarding traffic, whereas control logic and functionality are moved to an external entity known as the SDN controller (Nunes et al., 2014; Lara et al., 2014). The network intelligence is logically centralized in trusted software-based SDN controllers that provide an abstract view of underlying network resources. The abstraction of the flow broadly unifies the behavior of different SDN agents (Xia et al., 2014; Jarraya et al., 2014). Such distinguishing features make SDNs flexible, vendor agnostic, programmable, and cost effective, and create an innovative network environment (Lara et al., 2014; Macedo et al., 2015). Despite these remarkable features and the promising architecture of SDNs, market and industry observers are apprehensive about the security and dependability of SDNs (Ding et al., 2014; Jammal et al., 2014). Today, the security of the SDN presents a challenge and a key concern. However, security was not considered in the initial SDN design (Jarraya et al., 2014; Hakiri et al., 2014).

The architecture of SDN poses new external and internal threats (Kreutz et al., 2015; Hakiri et al., 2014; Scott-Hayward et al., 2013). The integrity and security of the SDN remain untested in the logical centralization of network intelligence (Jarraya et al., 2014; Scott-Hayward et al., 2013). The entire network may be compromised through the SDN controller, which may be a single point of failure and primary target. Furthermore, SDNs are more programmable than traditional networks, thereby rendering SDNs more vulnerable in terms of security (Ali et al., 2014). The abstraction of available flows at the SDN controller helps significantly in harvesting the intelligence of underlying resources. This knowledge base can be used for further attacks, exploitations, and, in particular, reprogramming of the entire network. Likewise, the southbound interface of SDN can be targeted by using a diverse set of denial-of-service (DoS) and side-channel attacks (Jammal et al., 2014). Moreover, SDN agents can be potentially targeted and injected with false flows. Cyber-attacks launched through SDNs can result in more devastating effects than those launched through simple networks. The STRIDE threat analysis methodology (Hernan et al., 2006) demonstrates the strength and analysis of the OpenFlow (OF) (McKeown et al., 2008) protocol, which is the first viable SDN technology. However, this analysis focused on the exploitation of DoS attacks and execution of information disclosure (Kloti et al., 2013). Similarly, the lack of transport layer security (TLS) at the southbound interface can also lead to DoS attacks, rule modification, and malicious rule insertion (Benton et al., 2013).

Despite the fact that security was not considered as part of the initial design, each layer/interface of the SDN has its own security implications and requirements (Kreutz et al., 2015; Ali et al., 2014). Consequently, the SDN essentially necessitates dynamic forensic remediation and robust policy frameworks. Security must be built as part of the SDN architecture and delivered as a service to ensure the privacy and integrity of all connected resources. SDN necessitates a simple, scalable, cost-effective, and efficient secure environment.

To the best of our knowledge, this work is the first to advocate an approach to achieve secure and dependable SDNs while comprehensively surveying, analyzing, and classifying the security vulnerabilities, attacks, and challenges of each SDN layer/interface together with 44 state-of-the-art security mechanisms of SDN. Moreover, this paper presents a thematic layered/interface taxonomy of SDN security vulnerabilities, attacks, and challenges. Although two survey papers (Scott-Hayward et al., 2013; Ali et al., 2014) were presented in the past, these works lack comprehensive thematic classifications and focus more on utilizing SDNs to secure different networks. The present paper complements our own short tutorial paper (Akhunzada et al., 2015) with the main contributions listed below.

- We devised a contemporary layered or interface taxonomy of the reported security vulnerabilities, attacks, and challenges of the SDN to illustrate the main categories of security implications that pertain to each SDN layer/interface. The paper also

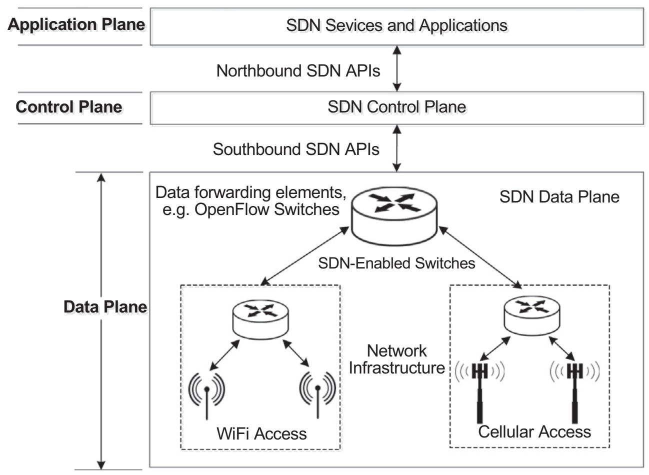

Fig. 1. A simplified view of the SDN architecture.

presents a broad overview of the existing security implications and challenges on each SDN layer/interface.

- We also highlight and analyze the possible threats that may affect and target a particular layer/interface alongside a suggested compact solution to help design secure SDNs.
- The ensuing paper also contributes by presenting state-of-the-art security solutions in SDN considering the earliest to the latest trends. The main categorization of solutions is followed by a critical analysis and discussion on devising a comprehensive thematic taxonomy. The critical discussion will extend the knowledge of the domain of the current security trends in the SDNs, the major strengths of potential SDNs, and the research gaps that need thorough investigations. The paper clearly differentiates and presents two main schools of thought in the SDN security domain (see Section 5.4).
- The paper analyzes each state-of-the-art security solution to identify the distinguishing SDN features utilized for each security solution and the exact problem addressed by a particular technique together with the simulation or emulation environment of the corresponding technique (see Table 3). The distinguishing features of SDN represent the potential of emerging SDNs
- The paper also identifies the potential effect of each state-of-the-art security solution on the corresponding SDN layers/interface (see Table 4).
- We also advocate the development of secure and dependable SDNs by presenting potential security requirements and their key enablers.
- Finally, we highlight the open issues, challenges, and future research directions that need to be addressed to develop secure and dependable SDNs.

The remainder of this paper is organized as follows: Section 2 introduces a simplified overview of the SDN architecture. In Section 3, the paper provides a broad overview of the security implications of each SDN layer/interface. We further classify broadly the reported security vulnerabilities, attacks, and challenges of SDN by devising a thematic layered/interface-based taxonomy. The possible threats that may affect and target a particular layer/interface with the suggested compact solution are highlighted and analyzed in Section 4. Section 5 presents state-of-the-art security solutions followed by a critical discussion to present a thematic classification. Moreover, the section contains the main categorization of state-of-the-art security mechanisms and identifies their potential effect on each SDN layer/interface. Furthermore, the section presents the employed approach to identify the distinguishing features of SDNs, addressed the exact problem with their implementation environment, and illustrates the two main schools of thought. Section 6 discusses the requirements and key enablers for dependable and secure SDNs. In Section 7, we highlight the open issues, challenges, and future research directions. Finally, we provide the concluding remarks in Section 8.

2. A simplified view of SDN architecture

The SDN architecture consists mainly of three planes, namely, application plane, control plane, and data plane, with their corresponding application programming interfaces (APIs) (Kreutz et al., 2015; Nunes et al., 2014). Figure 1 depicts a simplified view of the SDN architecture, which is explained by using a top-down approach. The simplified overview is provided to help the readers gain an easy and smooth understanding of the analysis of the SDNs security vulnerabilities, attacks, challenges, and state-of-the-art solutions.

2.1. *Application plane*

The application plane is also known as the application layer, which is responsible for providing a set of services and applications, such as intrusion detection system (IDS), intrusion prevention system (IPS), deep packet inspection, load balancers, security monitoring, and access controls (Xia et al., 2014; Jarraya et al., 2014). The SDN applications are basically programs that directly, explicitly, and programmatically share the anticipated network behavior and requirements with the SDN controller via northbound APIs. The applications and services can extract information with regard to the policy or behavior of the underlying architecture of SDNs. Furthermore, application-to-control plane communication is carried out because of various reasons (Xia et al., 2014; Ahmad et al., 2015; Shuja et al., 2014).

2.2. *Northbound SDN interface*

To support application or service orchestration, automation, and innovation, the SDNs employ open APIs, which are commonly known as northbound APIs. The northbound interface enables the application-to-control plane communication, and it is also recognized as the controller–service communication interface. Moreover, the interface facilitates in providing the abstract view of the underlying network. Likewise, the northbound APIs empower the direct expression of network behavior and requirements. However, the northbound SDN interface/APIs are more likely implemented on an ad hoc basis because no standard northbound SDN interface/APIs exist at present (Xia et al., 2014; Jarraya et al., 2014).

2.3. *Control plane*

The control plane is also referred to as the control layer of the SDN (Jarraya et al., 2014). The control plane includes a special network component called the SDN controller, which is logically centralized but physically distributed in principle (Kreutz et al., 2015; Fei et al., 2014; Xia et al., 2014). The controller is a software platform that is responsible for establishing and terminating flows and paths within SDNs. The SDN controller provides programmatic interfaces to the underlying network. The overall management functionality of SDN is simply entrusted in the SDN controller while it facilitates the programmability of the entire network. Likewise, the control layer also provides an abstraction of the underlying resources (Xia et al., 2014; Jarraya et al., 2014). Moreover, the SDN centralized logical control model can be applied to a wide variety of applications, underlying networks, and physical

Table 1
The SDN controllers with their description.

SDN controllers	*Open source*	*Language*	*Description*
NOX (Gude et al., 2008)	Yes	C++ – Python	NOX is the first controller written in C++/Python to support fast, asynchronous IO
POX (POX, 2014)	Yes	Python	Written in Python. Performs well as compared to NOX applications (especially when run under PyPy)
Maestro (Ng)	Yes	Java	Maestro provides the *view* abstraction for grouping related network state into a subset
Floodlight (Erickson, 2012)	Yes	Java	Floodlight can manage both OpenFlow and non-OpenFlow networks
Beacon (Erickson, 2013)	Yes	Java	Beacon is a cross-platform, modular, Java-based controller that supports both event-based and threaded operation
OpenDayLight (Medved et al., 2014)	Yes	Java	It uses OSGi framework and provide REST API having weak consistency.
Trema (Shimonishi et al., 2011)	Yes	Ruby/C	Trema is a full-stack, programming framework that allows users to develop and test OpenFlow controllers on a laptop
RouteFlow (RouteFlow, 2014)	Yes	C++	It is a special purpose controller and provides virtualized IP routing over OpenFlow hardware
Ryu (Ryu, 2014)	Yes	Python	It supports OpenFlow from version 1.0 to version 1.3 and integrates with Open-Stack, building virtual network without using VLAN
FlowVisor (Sherwood et al., 2009)	Yes	C	It is a special purpose controller for OpenFlow network virtualization.
SNAC (SNAC, 2014)	No	C++	A NOX-0.4 based controller to manage the network, configure devices and monitor different events
Helios (Shimonishi et al., 2010)	No	C	It provides a programmatic shell for performing integrated experiments
ONOS (Berde et al., 2014)	Yes	Java	Building networks for service provider with performance, scale-out design, and high availability

media, such as wired (e.g., Ethernet), wireless (e.g., 802.11 and 802.16), and optical networks (Jarraya et al., 2014; Ding et al., 2014; Baldini et al., 2012; Yoo, 2014; Pfeiffenberger and Jia Lei, 2014; Yang et al., 2014). Some popular SDN controllers and their corresponding brief descriptions are shown in Table 1.

2.4. Southbound interface/APIs

To support the overall programmatic control of the forwarding plane, event notification, capability advertisement, and statistics reporting, the SDN uses southbound interface/APIs (Farhady et al., 2015). The southbound interface/APIs provide a link between the control layer (control plane) and the infrastructure layer (data plane). Particularly, the southbound SDN interface/APIs enable communication between a controller and a switch. Thus, the interface is also known as a controller–switch communication interface. The interface assists the administrators in handling traffic of the underlying switching hardware of the data plane by pushing out controller decisions.

Currently, OF is the most popular and common southbound interface. People consider OF and SDN synonymous, although this is a misconception. In reality, OpenFlow represents a part of the entire SDN architecture (Lara et al., 2014). OF is a control-to-data plane communication protocol, and it is not the only existing protocol. Some SDN proprietary southbound protocols include Cisco's Open Network Environment Platform Kit and Juniper’s Contrail (Jarraya et al., 2014). An alternative to OF protocol is the forwarding and control element separation framework (Doria et al., 2010).

2.5. Data plane

The data plane is composed of underlying network infrastructures and is also known as the infrastructure layer of the SDN (Xia et al., 2014; Jain and Paul, 2013). The data plane consists of forwarding hardware, such as switches and routers. The control function is entrusted to the controller; thus, the underlying hardware, such as switches and routers, is responsible only for data forwarding and is also acknowledged as the forwarding plane of the SDN (Fei et al., 2014). The data plane implements the management functionality of the controller through SDN-enabled switches. Subsequently, the SDN-enabled switches are used to forward data, collect network information, and send the information back to the control plane via southbound interfaces (Govindarajan et al., 2013).

3. A layered/interface taxonomy of SDN security vulnerabilities, attacks and challenges

The section presents a comprehensive overview of each SDN layer/interface security vulnerabilities, attacks, and challenges. We further broadly classify the reported security vulnerabilities, attacks, and challenges of SDN by devising a thematic taxonomy based on each SDN layer/interface. The layered/interface taxonomy of SDN security vulnerabilities, attacks, and challenges is depicted in Fig. 2. The taxonomy clearly illustrates the main categories of security implications of each layer/interface.

3.1. Application plane security vulnerabilities, attacks and challenges

Controlling the network by using software is the principal property of SDNs. Thus, most implemented and deployed applications of SDN represent diverse network functions and can access the underlying network resources under certain privileges (Wen et al., 2013). SDN applications have many advantages, and yet they cause serious security challenges. This section briefly yet extensively explores the application plane-related security vulnerabilities, attacks, and challenges, which are classified into eight main categories, namely, (1) nested applications, (2) applications abusing SDN internal storage, (3) applications abusing SDN control messages, (4) trust establishment, (5) third-party applications and open development environments, (6) authentication, authorization, and accountability, (7) exhaustion of resources, and (8) application executing system commands.

1) Nested applications: Nested applications present a real challenge to deal with and are vulnerable to the following reported exploitations (Tsou et al., 2012; Monsanto et al., 2013):

 i) Service chain interference: Applications with chained execution may cause serious interference and security challenges. For instance, malign applications that participate in a service chain can drop control messages before the awaited applications, thus causing extreme interference. Moreover, interference may occur when a malicious application falls in an infinite loop to stop applications with chained execution.
 ii) Gateway to unauthorized access: A malevolent nested application can sidestep the access control by issuing the instance of another class application and can be a gateway to unauthorized access.

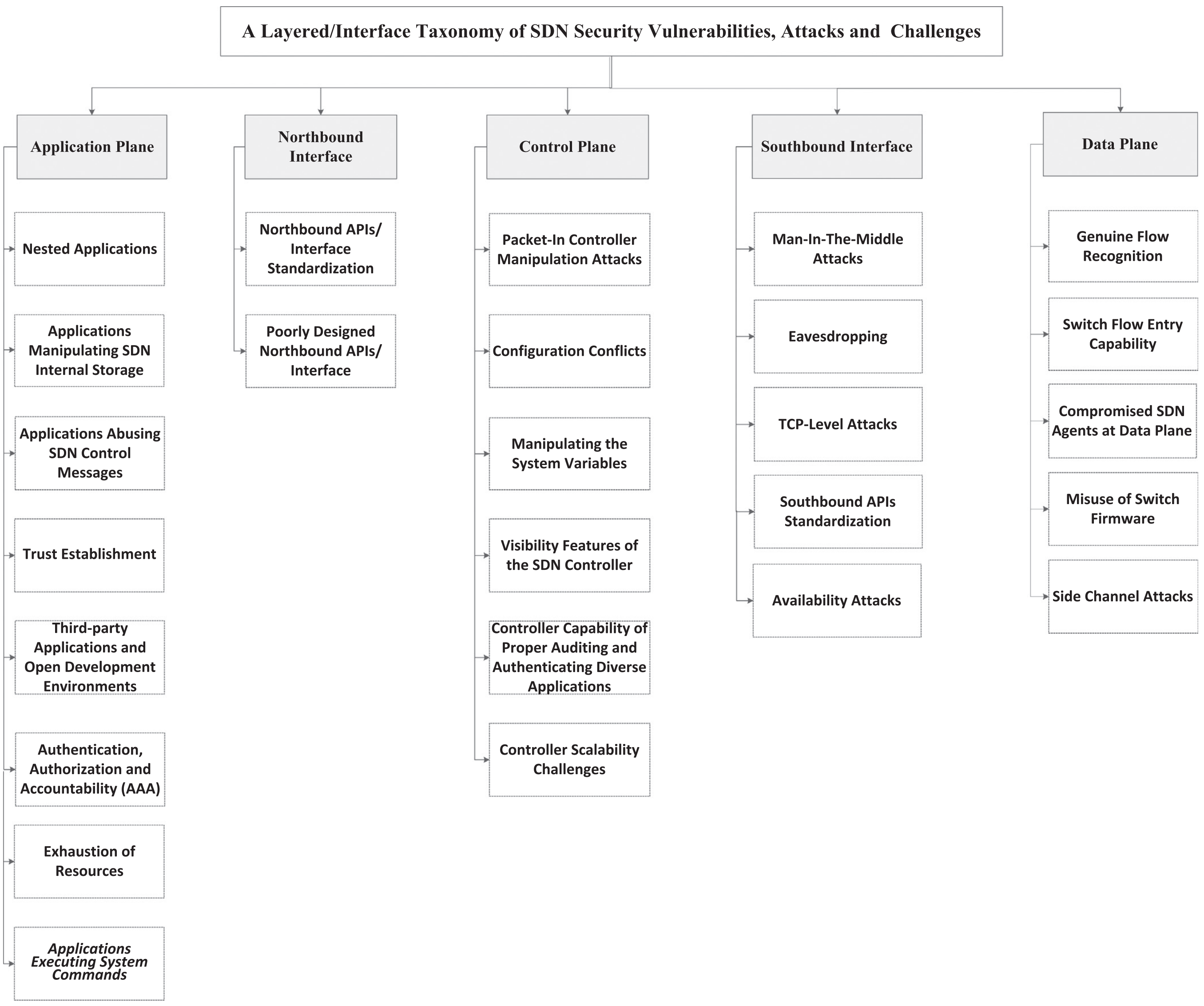
A Layered/Interface Taxonomy of SDN Security Vulnerabilities, Attacks and Challenges
Application Plane
Nested Applications
Applications Manipulating SDN Internal Storage
Applications Abusing SDN Control Messages
Trust Establishment
Third-party Applications and Open Development Environments
Authentication, Authorization and Accountability (AAA)
Exhaustion of Resources
Applications Executing System Commands
Northbound Interface
Northbound APIs/ Interface Standardization
Poorly Designed Northbound APIs/ Interface
Control Plane
Packet-In Controller Manipulation Attacks
Configuration Conflicts
Manipulating the System Variables
Visibility Features of the SDN Controller
Controller Capability of Proper Auditing and Authenticating Diverse Applications
Controller Scalability Challenges
Southbound Interface
Man-In-The-Middle Attacks
Eavesdropping
TCP-Level Attacks
Southbound APIs Standardization
Availability Attacks
Data Plane
Genuine Flow Recognition
Switch Flow Entry Capability
Compromised SDN Agents at Data Plane
Misuse of Switch Firmware
Side Channel Attacks

Fig. 2. A layered/interface taxonomy of SDN security vulnerabilities, attacks and challenges.

2) Applications manupulating SDN internal storage*:* The application in the application plane receives certain privileges to access the underlying resources; thus, the SDN controller shares internal storage among various SDN applications (Wen et al., 2013). Eventually, applications can access and manipulate the internal database of an SDN controller, which can further be used for many subsequent attacks, such as manipulating network behavior (Shin et al., 2014).
3) Applications abusing SDN control messages*:* A control message is responsible for the two-way communication between the data plane and application plane. An arbitrarily issued control message of SDN by an application may lead to the following attacks (Attacks; Dover):

   i) A malicious application that overwrites an existing flow rule in the controller switch flow table may lead to unexpected network behavior; this phenomenon is known as flow rule modification.
   ii) A malicious application may block all communication by issuing a control message that clears the flow table entries of an SDN switch.

4) Trust establishment: To establish trust between the SDN applications and the controller, compelling trust mechanisms must be present (Kreutz et al., 2013). An application server that stores sensitive user information can be compromised, and legitimate user credentials can subsequently be used to add forged but authorized flows to the network. Mechanisms to certify network devices exist. However, compelling mechanisms to establish trust to certify network applications do not exist. Moreover, the centralized control architecture of SDNs necessitates a centralized system to certify the multitude and diversity of network applications and presents an interesting area that is yet to be explored.
5) Third-party applications and open development environments: Third-party applications could also result in serious security vulnerabilities and challenges because of the lack of standard and consensus-based development environments, programming models, and paradigms, and the variety of vendors (Tsou et al., 2012; Shin et al., 2013). Importantly, third-party applications could cause serious issues of interoperability and collision in security policies. Moreover, dealing properly with the diversity and multitude of third-party

applications and non-standard open software development environments is challenging.

6) Authentication, authorization and accountability (AAA): Authentication of nested applications is a major challenge in programmable networks, and the diversity of third-party applications makes this situation difficult. Authorization-related attacks can lead to illegal access to the controller, thereby affecting the lower three corresponding layers/interface (Kreutz et al., 2013). In-authentic applications can damage the application layer, northbound interface, and control layer. No compelling authorization mechanisms exist for such application in the centralized control architecture of SDN with open software development environments. Likewise, accountability is another real challenge considering the various third-party and nested applications responsible for the consumption of network resources (Tsou et al., 2012; Kreutz et al., 2013).
7) Exhaustion of resources: Malicious applications can exclusively contribute to exhaust all the available system resources and seriously affect the performance of other applications, including the SDN controller. Such attacks have been verified (Wen et al., 2013).

   i) Memory consumption: A malicious application may be involved in continuous consumption of system memory or in memory allocation to exhaust all the available system memory.
   ii) CPU consumption: A malicious application can seriously exhaust all the available CPU resources by simply creating useless working threads.
   iii) A malicious application can execute a system exit command to dismiss the controller instance.

8) *Application executing system commands:* A malicious SDN application is capable of terminating the controller instance by executing a system exit command. The attack was demonstrated previously (Shin et al., 2014; Attacks).
9) *Critical remarks:* The security vulnerabilities, attacks, and challenges discussed above are critical and can target all the corresponding SDN layers/interface. Protection against diverse malign applications will remain a challenge for SDNs (Zhang, 2013). No compelling mechanism for distinguishing user or third-party or network service applications exists, and the accountability and access control of nested applications have not been demonstrated.

3.2. Northbound APIs/interface security vulnerabilities, attacks and challenges

Application plane has direct implications on the overall underlying SDN architecture. Hence, the northbound APIs are considered highly important targets for exploitation. The reported security vulnerabilities, attacks, and challenges of the northbound interface are stated below. Two main challenges cause the vulnerability of northbound APIs.

1) Northbound APIs/interface standardization*:* No standard northbound API exists, and working with various northbound open APIs is challenging (Akhunzada et al., 2015; Pan and Nadeau, 2011). Moreover, open independent development environments without standard specifications are faced with increased risk from various security challenges posed by skilled adversaries.
2) Poorly designed northbound APIs/interface: A poorly designed northbound interface can be misused easily by SDN applications to manipulate the behavior of other applications (Kreutz et al., 2013). For instance, an SDN application may exploit a poorly designed northbound API to evict an ongoing application session. Moreover, an SDN application may use a poorly designed northbound interface to randomly unsubscribe a target application, thereby rendering it incapable of obtaining important subscribed control messages that can be easily carried out by the unsubscription of an event listener (Attacks).

3.3. Control plane security vulnerabilities, attacks and challenges

The distinguishing property of SDNs is centralized control architecture, which results in significant managerial benefits. However, this property represents a single point of failure. The norm of SDNs is the centralized control network intelligence. Moreover, the SDN controller has a pivotal role in management, and it is a solo decision making entity, thus becoming a primary target. Furthermore, the visibility features of the SDN controller also pose serious security challenges. The section elaborates control plane-related security vulnerabilities, attacks, and challenges.

1) Packet-in controller manipulation attacks: A packet-in essentially represents a packet that does not match any flow rules at the data plane, and the OF protocol mandates that such packets must be sent by the switch to the controller directly. When a message that is received by an SDN switch with no match entry is directed to a controller by the OF protocol by default, such messages are called packet-ins. At present, the control plane has no built-in security mechanism to avoid the manipulation of packet-in messages even if the OF is TLS enabled. Authorized switches can also send forged packet-in messages that can subsequently be used to corrupt the controller state by the following practical attacks that may occur on many SDN controllers (Hong et al., 2015; Dhawan et al., 2015):

   i) Directed DoS attacks: An attacker may place the controller in an unpredictable state by flooding a target SDN controller with packet-ins; this attack was demonstrated in a previous work (Dhawan et al., 2015). One of the practical forms of launching a directed DoS attack is packet-in flooding, which places the SDN controller in an unpredictable condition. Major SDN controllers are still a target of this attack. DoS attacks using packet-ins may be carried out in many ways. A DoS attack is a serious concern in the centralized control architecture of SDNs.
   ii) Poisoning the view of the controller: An address resolution protocol (ARP) packet relayed as a packet-in message can be forged to adversely affect the view of the controller; this attack was demonstrated in a previous work (Hong et al., 2015).
   iii) Poisoning the network topologies and traffic hijacking*:* Manipulating the link discovery service relayed as packet-in messages can be utilized to create fake network topologies. Moreover, the vulnerability of the host tracking service can be exploited to establish traffic hijacking (Dhawan et al., 2015).
   iv) Fabricated links creation: An LLDP packet relayed as a packet-in message may also be forged to create a fabricated link (Dhawan et al., 2015).
   v) Side-channel attacks: Manipulating packet-ins can also leverage side-channel attacks to obtain sensitive information (Dhawan et al., 2015).
   vi) Other challenges and vulnerabilities of control messages: Control messages can also be manipulated to corrupt the state of the controller in many ways. Some practical attacks and vulnerabilities of control messages include spoofing a target switch and the attack (Attacks), and switch-table flooding attack using Data Path ID to place the controller in an unpredictable state (Attacks).

2) Configuration conflicts: The control plane enforces a network-wide policy. However, single-domain multiple SDN controllers,

multi-tenant SDN controllers, and multiple OF architectures may lead to serious configuration conflicts and inter-federated configuration conflicts (Al-Shaer and Al-Haj, 2010). Moreover, stateful applications may not work properly because the controller may not be able to synchronize the network updates, and it does not have the capability to recall past events.

3) Manipulating the system variables: Manipulating a system variable may also corrupt the controller state. For instance, a system time alteration by an attacker may turn the controller practically off from linked switches (Attacks).
4) Visibility features of the SDN controller: The visibility features of SDN can also be vulnerable to harvesting of network intelligence of the underlying architecture for further exploitation (Akhunzada et al., 2015).
5) Controller capability of proper auditing and authenticating diverse applications: The security of the control plane is normally measured and challenged in terms of controller capabilities. For instance, a real challenge is enabling the controller to properly facilitate the authentication and authorization of network resources consumed by applications implemented on top of the control plane with appropriate tracking, auditing, and isolation (Hartman et al., 2013; Sezer et al., 2013).
6) Controller scalability challenges: The controller is the sole entity responsible for centralized decision making. Controllers nowadays become a bottleneck in a 10 Gbps link high-speed network. Moreover, the lack of scalability causes the following serious problems detailed in the literature (Sezer et al., 2013):

i) Saturation attacks
ii) Experience delay constraints
iii) A single point of failure

3.4. Southbound API/interface security vulnerabilities, attacks and challenges

Separating the control plane and data plane results in serious disasters; more importantly, control plane security has direct implications on the data plane (Kreutz et al., 2015). Thus, the controller–switch communication channel remains a favorable choice for attackers. The specification of Open-Flow (OF) datagram transport layer security and TLS for channel security is optional in the latest versions of OF (Kreutz et al., 2013). However, TLS is not secure from TCP-level security attacks (Liyanage and Gurtov, 2012) and is not reliable (Kreutz et al., 2013) in many cases. Subsequently, the diverse attacks described below occur.

1) Man-in-the-middle attacks: The southbound interface provides an opportunity to actively control the control channel, thereby causing man-in-the-middle attacks. In this type of attack, a skilled attacker modifies the control messages exchanged between the control plane and the data plane, such as flow rule messages, to corrupt network behavior (Benton et al., 2013; Attacks).
2) Eavesdropping: The southbound interface can be targeted for both active and passive eavesdropping. For instance, an opponent may steal highly sensitive information by sniffing the control channel, i.e., learning about the advertised network topologies by sniffing ongoing control messages (Attacks).
3) TCP-level attacks: The southbound APIs can be targeted for TCP-level attacks because the optional TLS does not implement TCP-level protection; the attacks were demonstrated in a previous study (Liyanage and Gurtov, 2012).
4) Southbound API standardization: Unlike the northbound APIs, no standard southbound API exists, which presents a real challenge for the SDN community. A compelling and consensus-based southbound interface is needed to address the security vulnerabilities, attacks, and challenges of the interface thoroughly and uniformly (Kreutz et al., 2015).
5) Availability attacks: Availability-related attacks refer to DoS and distributed denial of service (DDoS) attacks. The south-bound interface can be targeted by communication flooding attacks, which specifically affect the control layer, the control data interface, and the data layer (Scott-Hayward et al., 2013; Benton et al., 2013).

3.5. Data plane security vulnerabilities, attacks and challenges

The SDN switches and routers are dump forwarding devices. The capability of controller decisions is based on these data plane forwarding entities. This section explains some of the foremost data plane-specific security vulnerabilities, attacks, and challenges.

1) Genuine flow recognition: The controller decisions are based on the SDN switch flow rules. Hence, one of the key challenges for the controller is recognizing and differentiating whether the switch-generated flow rules are candid or fraudulent. A compromised switch that generates malicious flow rules can render the data plane practically offline (SNAC, 2014).
2) Switch flow entry capability: An SDN switch has a limited ability to maintain the number of flow entries, and an SDN switch can practically be targeted with saturation attacks. These attacks subsequently lead to switch DoS attacks that render the data plane in an unpredictable state. For instance, an attacker can simply install or flood a large number of flow entries to exhaust the switch-limited resources. Flow rule flooding was demonstrated in the literature (Sherwood et al., 2009; Monsanto et al., 2013; Shin et al., 2014).
3) Compromised SDN agents at data plane: A compromised host or SDN switch can contribute to a variety of attacks. For instance, an attacker can fill up the compromised target switch flow table to advertise fake topologies (logical or physical) or render the controller in an unpredictable state. These attacks were demonstrated in previous works (Farhady et al., 2015; Doria et al., 2010). Compromised SDN agents can also lead to a variety of man-at-the-end (MATE) attacks. Furthermore, a manipulated control message by an attacker can render the data plane in an unpredictable state and disconnect the control plane from the data plane. A compromised host or SDN switch may also trigger dynamic attacks, such as traffic rerouting, traffic hijacking, and network DoS attacks.
4) Misuse of switch firmware: A previous study indicated that certain SDN switch hardware tables cannot process crafted flow rules (Ryu, 2014).
5) Side-channel attacks: The data plane can be attacked with side-channel attacks. For example, an input buffer can be used to identify flow rules, and analyzing the packet processing time may determine the forwarding policy (Scott-Hayward et al., 2013).

4. Possible security threats affecting each SDN layer/interface

This section presents and analyzes the possible security threats within SDNs. Table 2 illustrates the analysis on a particular security threat that affects each SDN layer/interface. Security threats that affect corresponding layers/interface must be addressed in emerging SDNs. The table also presents the protection mechanisms, security requirements with their corresponding affected functionalities, and SDN layers/interface against each possible threat. Operating system (OS) alteration represents the destruction or alteration of entire SDN elements, such as controllers (Akhunzada et al., 2015; Li et al., 2009). A threat can be

Table 2
Possible security threats affecting corresponding layers/interface of SDNs.

Possible security threats	***Protection techniques***	***Security requirements***	***Affected functionalities***	***Application layer***	***Northbound interface***	***Control layer***	***Southbound interface***	***Data layer***
Operating system alteration	Trusted computing	System integrity protection	Application management	✓		✓		✓
Software framework alteration	Trusted computing	System integrity protection	Application management	✓		✓		✓
Software failure	High assurance	Robustness, system integrity protection	All functionalities	✓		✓		✓
Hardware failure	High assurance	Robustness, system integrity protection	All functionalities			✓		✓
Configuration data alteration	Data integrity functionality in SDN middleware	Data integrity protection	Resource management, application management.			✓	✓	✓
Configuration data extraction	Data integrity functionality in SDN	Confidentiality protection	Data management			✓	✓	✓
Unauthorized access to SDN services	Deploying secure administration module	Identities verification, ensuring system integrity	All functionalities	✓	✓	✓	✓	✓
User data alteration	Data integrity functionality in SDN	Ensuring data integrity	Data management					✓
Masquerading as authorized SDN controller	Use of digital signatures for SDN software modules	Ensuring system integrity, identities verification, accountability	Application management			✓	✓	✓

mitigated by ensuring that system integrity is protected by implementing trusted computing. OS alteration can target all the layers of SDN and affect the management of running services and applications.

Software framework alteration identifies the destruction or alteration of a middleware or its components. Similar to securing against OS alteration, software framework alteration can be prevented by ensuring system integrity while implementing trust computing (Baldini et al., 2012). Furthermore, software framework alteration can affect all the layers of SDNs. Likewise, the software failure threat represents a general failure in any of the components that constitute the software framework, application, and operating system (Sakaguchi et al., 2003). Software failure can be mitigated by employing high-assurance techniques while ensuring the robustness of a system. Unlike the software framework threat, software failure can affect all layers of SDNs.

Hardware failure represents generic failure of a hardware in any component (Baldini et al., 2012). Similar to software failure, hardware failure can be reduced by employing high-assurance techniques while ensuring the robustness of a system. The entire set of functionalities is affected by the hardware failure threat that targets the control and data layer.

The configuration data alteration threat represents the destruction or alteration of configuration data that are required by SDNs to perform different functions (Li et al., 2009). Configuration data can be removed or modified. The threat can be mitigated by ensuring data integrity in the SDN middleware. Furthermore, the threat can target the control layer, control–data interface, and data layer while possibly affecting the resource and application management functionalities. Similarly, the configuration data extraction threat is an eavesdropping threat (Garnaev and Trappe, 2014) where the attacker gathers configuration information that can be utilized in subsequent attacks. The threat can target the control layer, control–data interface, and data layer.

Unauthorized access to SDN services is identified as a security breach (Sakaguchi et al., 2003). The threat can be mitigated by deploying secure administration modules while ensuring system integrity. The security requirements for mitigating the threat are identification and verification. However, the threat can affect the entire set of functionalities and target all the layers/interfaces of SDN.

User data alteration threat involves the destruction or alteration of user data, such as customized profiles of user traffic (Baldini et al., 2012; Li et al., 2009). The user data alteration threat can be mitigated by ensuring data integrity. The threat can affect data management and target the data layer. By masquerading as an authorized SDN controller, the threat identifies the activation of a malicious software on SDNs, such as a controller platform (Baldini et al., 2012). The threat can be mitigated by using the digital signatures for SDN software modules. Threat mitigation requires system integrity, identity verification, and accountability. Application management can be affected by the threat activation and target the control layer, control–data interface, and data layer of the SDN.

5. State-of-the-art SDN Security solutions: a complete analysis and overview

In this section, we provide a complete analysis and overview of existing state-of-the-art SDN security solutions. Basically, we present the main classification of the innovative security solutions of SDNs, as shown in Fig. 3. SDN security solutions are classified into six main categories, namely, (a) secure design of SDN, (b) implementation of satisfactory audit, (c) enforcement of security policy, (d) security monitoring and analysis, (e) security

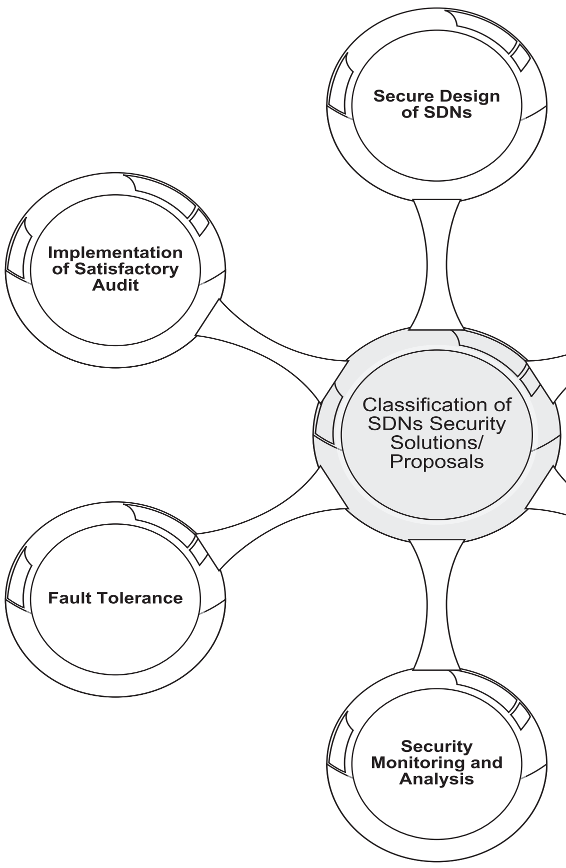
Secure Design of SDNs
Implementation of Satisfactory Audit
Enforcement of Security Policy
Classification of SDNs Security Solutions/ Proposals
Fault Tolerance
Security Augmentation
Security Monitoring and Analysis

Fig. 3. Main classification of the state-of-the-art SDNs security solutions.

augmentation, and (f) fault tolerance. We further classify the surveyed solutions by devising a thematic taxonomy based on the SDN layers/interfaces, the distinguishing features of SDNs, implementation environment, and security objectives. The taxonomy is presented in Fig. 4. This section also presents the identification of the distinguishing features and the effect of security solutions on each layer/interface of the SDN.

5.1. *Secure design of SDN*

Efforts in this category are limited. FRESCO (Shin et al., 2013) a security-specific application development framework for OF networks, was proposed in the literature. FRESCO is a security application development platform that facilitates the exportation of API scripts, which help security experts develop threat detection logic and security monitoring as programming libraries. Moreover, FRESCO is a click-inspired (Shin et al., 2013) programming framework that facilitates the rapid design and modular composition of different security mitigation and detection modules using OF. The implementation and evaluation demonstration of FRESCO is performed through NOX (Gude et al., 2008), which is an open-source OF controller; however, the security constraints generated by these applications are controlled and enforced by FortNox (Porras et al., 2012). FRESCO work can be extended to different architectures (Cai et al., 2010; Mogul et al., 2010).

FortNox (Porras et al., 2012) employs a security enforcement kernel (SEK) to enforce flow constraints for active defense against different threats. FortNox is basically an enforcement engine that is responsible for avoiding rule conflicts from different security authorizations. FortNox uses two protection mechanisms: (1) rule prioritization, which ensures that any new rule that contradicts the rules produced by FRESCO applications are simply overridden because of the highest priority, and (2) the conflict detection algorithm is applied to each new rule, thus rejecting any new rule immediately when a conflict is detected. FortNox is merely a software extension that deals mainly with particular OF application policy violation, dynamic flow tunneling, and rule conflict detection. Furthermore, it addresses flow rule contradiction in real time and avoids any adversarial OF application attempts to inject flow rules to bypass SEK enforced rules. Work on FortNox is highly inspired by previous research (Al-Shaer and Al-Haj, 2010; Al-Shaer et al., 2009; El-Atawy et al., 2007; Liu, 2008; Xie et al., 2005); the FRESCO work is an extension carried out by the authors of FortNox.

Remarks. *:* The first proposal is a major contribution toward secure programming, which has a direct effect on the application layer, control layer, and interfaces between the two layers, except the data layer. The second proposal focuses on rule conflict and authorization, which primarily affect the control layer and south and northbound interfaces. However, this action does not improve the security of the application layer and infrastructure layer. FortNox is classified in this category mainly because of its SEK that is used for real-time verification.

5.2. *Implementation of satisfactory audit*

Security audit is an important area that must be addressed thoroughly. Verificare (Skowyra et al., 2013) was proposed as a design and modeling tool that satisfies all the requirements of a system design, and it is a key contribution in this category. Verificare is an audit design and modeling tool for verifying real-world system properties against a system model, which highlights and traces any property violation. Moreover, the tool guarantees network safety, correctness, and reliability. Verificare is designed

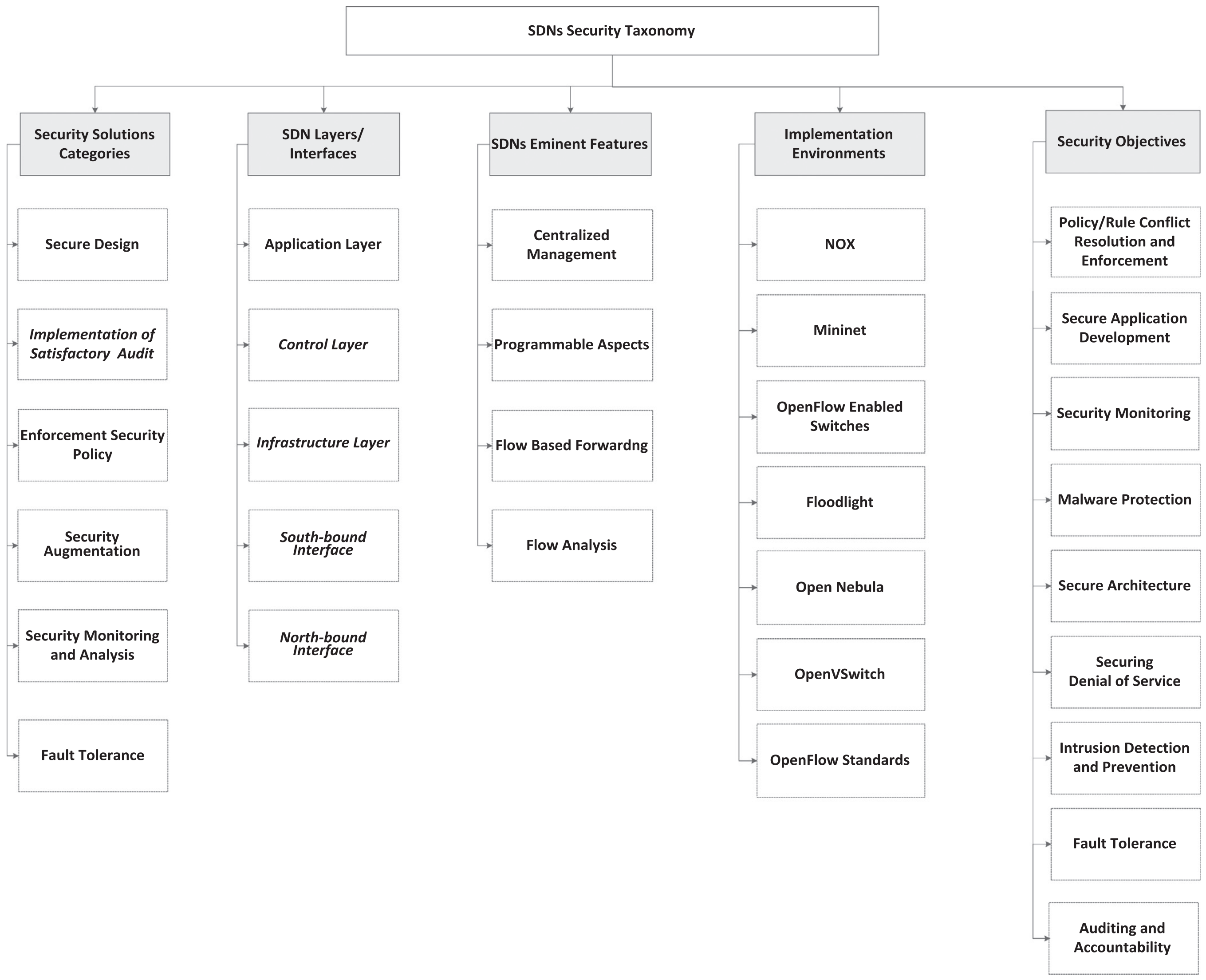

Fig. 4. Taxonomy of SDNs security solutions.

using a complementary pair of SDNs and formal verification. The authors further provide an example for their work using OF-based learning switches to enable communication between mobile nodes. This particular proposal also considers the verification of network correctness and specification modeling while considering scalability issues. Some intersecting verification tools are available, such as PRISM (Kwiatkowska et al., 2011) and Proverif (Blanchet, 2005); however, these tools are not completely comparable to Verificare.

Another major contribution in this area (Handigol et al., 2012) allows software developers of SDNs to trace the root cause of bugs by reconstructing a series of events that cause that particular bug. Packet back-trace assists SDN programmers in resolving logical errors, helps implementers of the switch to resolve protocol compatibility errors, and helps network operators in submitting a complete bug report to vendors. This tool also uses a network debugger, which is a programmatic control used to facilitate new ways of debugging networks. For instance, in debugging the network, servers could not connect to clients and no forwarding rule for packet matching is found in the middle of the network; this approach is similar to the detection of servers placed in the wrong location. Moreover, this tool effectively tracks the root cause of the bug.

The virtual source address validation edge (VAVE) (Yao et al., 2011) is a significant contribution in this area and is used to solve the problem of source address validation. VAVE also checks whether the source of the packet is valid or not based on the generated rules. The VAVE application is designed mainly to validate the source address. However, spoofing is still a problem on the Internet. Another standard called source address validation (SAVI) (Saucez et al., 2013) does not provide complete protection against spoofing because of solution space constraints; VAVE can be employed to improve SAVI (Yao et al., 2011). Moreover, VAVE can be used to avoid attacks related to IP spoofing, SYN flooding, smurf attacks, and DNS amplification (Yao et al., 2011). Furthermore, VAVE uses NOX controller to regulate the validation rules from a global view on each SAVI device.

A more recent contribution in the area of security audit is Fleet (Matsumoto et al., 2014). Fleet defends against a malicious administrator who is attempting to reprogram the entire network. Fleet is useful against damage to routing and forwarding and disturbance of network availability by misconfiguring the SDN controller.

Table 3 summarizes the state-of-the-art SDN security solutions. The table demonstrates clearly the distinguishing features utilized for each state-of-the-art SDNs and the exact problem addressed by a particular technique together with the simulation/emulation

Table 3
Classification and comparison of the State-of-the-art SDN security solutions.

Classification	***Techniques***	***SDNs features***	***Problem addressed***	***Simulation/emulation environment***
Secure design	FRESCO (Shin et al., 2013)	Centralized management	Secure application development and modular composition of detection and mitigation security modules.	NOX
	FortNOX (Porras et al., 2012)	Flow based forwarding and dynamic updation of rules	The problem of rule conflict and provisioning of role-based authorization.	NOX
Security audit	VERIFICARE (Skowyra et al., 2013)	Programmable aspects at control plane	Verification of network correctness and specification modeling	OpenFlow
	SDN-Debugger (ndb) (Handigol et al., 2012)	Flow based forwarding and programmable aspects	Tracing the root cause of bugs in a Network	Mininet
	Fleet (Matsumoto et al., 2014)	Centralized management	Malicious administrator problem	Mininet and POX (Fleet Controller)
	VAVE (Yao et al., 2011)	Traffic analysis and dynamic updation of rules	Source address validation	NOX
Security enforcement policy	FLOVER (Son et al., 2013)	Flow rules policies	Model checking for security policies to do real time verification	NOX
	Flow-based Security Language (FSL) (Hinrichs et al., 2008)	Flow control features of SDNs	Network security policy enforcement	NOX
	Splendid Isolation: language based security (Schlesinger et al., 2012)	Centralized management	Traffic isolation and slice abstraction	OpenFlow
	No bugs In Controller Execution (NICE) (Canini et al., 2012)	Programmable Aspects (controller programs)	Policy for exclusion of bugs in Controller at application run time using finite-state model checking	NOX
	LiveSec (Wang et al., 2012)	Flow control and an introduction of new Access-Switching layer	Interactive Policy Enforcement	NOX
	SANE (Casado et al., 2006)	Centralized management	Central Policy Enforcement	SANE Controller
	Ethane (Casado et al., 2007)	Centralized Control and dynamic updation of rules	Flow-rule Enforcement	Ethane Controller
	PermOF (Wen et al., 2013)	Centralized management	Policy enforcement of Access Control	OpenFlow
	FLOWGUARD (Hu et al., 2014)	Flow path analysis and dynamic updation of rules	Verification of Security Policy	Floodlight
	SDNs firewall using Header Space Analysis (HSA) (Wang et al., 2013)	Flow space analysis and dynamic updation of rules	Developing a robust firewall application for SDNs	Floodlight
	VeriFlow (Khurshid et al., 2012)	Centralized management, flow analysis and dynamic updation of rules	Verification of network-wide invariants in real time using equivalence classes and forwarding graphs	Mininet
	NetPlumber (Kazemian et al., 2013)	Centralized management	Policy Compliance checking in real-time	Google's SDN, the Stanford backbone and Internet 2
	FlowChecker (Al-Shaer and Al-Haj, 2010)	Flow based forwarding and centralized management	Policy enforcement for flow-table configuration verification at application run time using binary decision graphs model checking.	GENI OpenFlow environment
Security augmentation	Slick Architecture (Anwer et al., 2013)	Centralized control and flow analysis	Traffic steering based on prerequisite security criteria (Policy Enforcement)	Slick Controller
	FlowTags Architecture (Fayazbakhsh et al., 2013)	Centralized control	An extended SDNs architecture for Correctness of systematic policy enforcement.	SDN controller
	OrchSec (Zaalouk et al., 2014)	Traffic monitoring and centralized management	Controller-agnostic security application development	Mininet, Floodlight and POX
	Covert channel protection using Filter (Liu et al., 2011)	Centralized control and packet analysis	The problem of Covert Channel attacks	OpenFlow
	SIMPLE (Qazi et al., 2013)	Centralized control	middlebox-specific traffic steering	Extended POX (SIMPLE controller)
	OF-RHM (Jafarian et al., 2012)	Centralized control and dynamic updation of rules	Moving target defense	NOX

Table 3 (*continued*)

Classification	***Techniques***	***SDNs features***	***Problem addressed***	***Simulation/emulation environment***
	Self-Organizing Maps (SOM) (Braga et al., 2010)	Traffic analysis and dynamic updation of rules	The problem of DDoS attacks	NOX
	OpenSAFE (Ballard et al., 2010)	Traffic analysis and centralized management	How to and where to route the traffic for network analysis and security monitoring applications	NOX
	ident++ (Naous et al., 2009)	Flow based forwarding and centralized management	Delegation of network security to end hosts for more flexible and powerful enforcement of policies.	ident++ controller (OpenFlow)
	Resonance (Nayak et al., 2009)	Centralized control with continuous monitoring	Access control enforcement for enterprise networks	
	FlowNAC (Jon Matias et al., 2014)	Traffic analysis and centralized management	Network access control	NOX
	MAPPER (Sapio)	Programmable aspects (Programmable switches and routers)	Fine grained access control policy enforcement	
	NetFPGA based solution (Goodney et al., 2010)	Programmable aspects of SDNs	Intrusion detection	
	ADS Revisit (Mehdi et al., 2011)	Programmable aspects of SDNs	An extension for finding anomalies detection accuracy	NOX
	CloudWatcher (Shin and Gu, 2012)	Flow monitoring and analysis	An extension for Cloud monitoring	NOX
	L-IDS (Skowyra et al., 2013)	Centralized management and dynamic flow rules	Intrusion detection	L-IDS controller and OpenFlow
	Network Intrusion detection and Counter-measure sElection (NICE) (Chung et al., 2013)	SDNs programmable switches and APIs	Intrusion detection and mitigation	OpenFlow
	SnortFlow (Xing et al., 2013)	Flow based forwarding	Intrusion Prevention	Snort and OpenFlow
Security monitoring and analysis	OpenWatch (Zhang, 2013)	Flows monitoring and analysis (Adaptive flow counting)	Load balancing and achieving anomaly detection accuracy	
	FleXam (Shirali-Shahreza and Ganjali, 2013)	Centralized management	Deep packet inspection (DPI)	OpenFlow
	AVANT-GUARD (Shin et al., 2013)	Data-to-control plane communication	The problem of DoS attack	POX
	NetFuse (Wang et al., 2013)	Flow analysis and centralized management	Protection against traffic over load caused by internal or external threats	OpenFlow
	CONA (Choi)	Traffic analysis and dynamic updation of rules	Resource exhaustive attacks	NetFPGA-OF (OpenFlow)
Fault tolerance	SPARC (Sharma et al., 2011)	Centralized management	Fault tolerance	NOX
	Link failure (Staessens et al., 2011)	Centralized management	Fault tolerance	NOX

environment of the corresponding technique. The table also shows that certain techniques with in-common classification exist. By contrast, techniques of a similar class are not exclusively of the same nature. The identification of features in the table represents the potential of emerging SDNs.

5.3. *Enforcement of security policy*

Enforcing security policy is a fairly important and serious issue in SDNs. The security research community of SDNs has paid considerable attention to addressing the issue of policy conflict resolution and enforcement. Researchers (Son et al., 2013) proposed FLOVER, which is a model-checking system that verifies the flow policies against the security policies of a network. FLOVER is a flow verification tool that detects inconsistencies that arise in a flow table against the security policy of the network by using satisfiability modulo theories solver. The performance of FLOVER is efficient. However, FLOVER is inspired by former research on modeling security policies of firewalls, and it does not address the dynamic flow rules in SDNs.

The other major contribution is No bugs In Controller Execution (NICE) (Canini et al., 2012). The NICE tool combines model checking with symbolic execution to determine inconsistencies in multiple OF applications. NICE proficiently uncovers bugs in OF programs while considering the correctness of the application installed on multiple devices. Another technique (Kothari et al., 2011) employed symbolic execution with the same principle as NICE to detect network protocol manipulation attacks.

FlowChecker (Al-Shaer and Al-Haj, 2010) adopts binary decision diagram (BDD) to handle intra-switch misconfiguration in a particular flow table. Flowchecker is used to resolve the conflicts that may arise across several OF switches. It simply encodes the flow table configuration through BDD and utilizes different model checking techniques to model the interconnected network of OF switches.

The VeriFlow scheme verifies real-time invariants (Khurshid et al., 2012). VeriFlow monitors dynamic changes in the network by constructing a model of the network behavior. Moreover, VeriFlow employs custom algorithms to derive network errors automatically. NetPlumber is a scheme that is similar to VeriFlow (Kazemian et al., 2013). However, NetPlumber employs header space analysis (HSA) and dependency graphs for real policy checking. Both VeriFlow and NetPlumber are protocol independent and have the same runtime performance. Both schemes support verification of forwarding actions in real time. Unlike VeriFlow, however, NetPlumber verifies arbitrary header modifications, including encapsulation and rewriting.

Remarks. : Several studies in this category seem almost similar and have almost the same implicit goal; however, these studies use explicitly different systems. A detailed technical comparison is shown in Table 4.

An important contribution (Hinrichs et al., 2008) presented a flow-based policy enforcement grounded on flow security language (FSL). Implemented on NOX controller, FSL employs the concept of network flow and handles policy conflicts between administrators and conflicts that arise in the rule set of the administrators. Moreover, the decision with regard to policy is enforced on each packet and on network links to hasten enforcement decisions per second. Another close category of language-based security is termed splendid isolation (Schlesinger et al., 2012). The authors discussed mainly the verification of the isolation of the program traffic. Moreover, the study addressed the problem of programming networks securely and reliably. Slice-based network programming abstraction and isolation protect the programs from outside interference with other important security benefits. In addition, it simplifies construction of programs.

A key contribution is LiveSec (Wang et al., 2012), which is a flexible and scalable security management architecture (Koponen et al., 2010). The main purpose of LiveSec is interactive policy enforcement to ensure complete end-to-end traffic control, distributed load balancing of security workload, and application awareness monitoring through live traffic monitoring and historical traffic replay.

SANE (Casado et al., 2006) protection architecture proposed a logically centralized controller for all routing and access control decisions and policies. The SANE proposal was considered a radical change to the network infrastructure. A significant extension of SANE is Ethane (Casado et al., 2007). Ethane primarily proposes two components for controlling the network: (a) a centralized logical controller to enforce global policy and (b) switches that forward packets according to flow table rules. However, Ethane was proposed for flow rule enforcement. SANE and Ethane established the foundation for emerging SDNs.

A significant contribution is PermOF (Wen et al., 2013), a fine-grained permission system that comprises a set of OF-specific permissions and runtime isolation mechanism for applying the permissions. The set of OF-specific permissions is designed considering four different aspects: (a) threat model, (b) controller implementation API set, (c) application functional requirements, and (d) control messages in OF. The proposed isolation mechanism isolates the controller and applications in a thread container. The applications cannot call controller procedures or directly refer to the memory of a kernel. Moreover, the application and OS are isolated by introducing a shim layer between them called an access control layer. The shim layer is controlled by the kernel of a controller. Basically, the design consideration is setting of permission and isolation mechanisms to enforce permission control.

FLOWGUARD (Hu et al., 2014) is a comprehensive framework responsible for the detection and resolution of policy violations of firewalls in the dynamic environment of SDNs. This technique helps in resolving policy conflicts automatically and in real time. The authors in (Wang et al., 2013) proposed a systematic approach to detect and resolve conflicts in SDN firewalls by checking the firewall authorization space and flow space. The main aim of the study was to build a robust SDN firewall. The proposed technique was used to implement a cross-check on both flow tables and firewall policies to detect conflicts efficiently. The approach searches the flow paths in the entire network, checks these paths against all firewalls, and denies rules to determine whether conflict arises with the firewall deny rules. However, the conflict

Table 4
Illustration of a technical comparison among the close categories of SDNs policy enforcement solutions.

Solutions (enforcement of security policy)	*Employed technique*	*Exact usage*
FLOVER	Satisfiability Modulo Theories (SMT) solver	Real-time
VeriFlow	Custom algorithms based on equivalence classes and forwarding graphs	Real-time
NetPlumber	Header Space Analysis and dependency graphs	Real-time
NICE	Finite-state model checking	Application run-time
FlowChecker	Binary Decision Diagram (BDD) model checking	Application run-time

resolution strategies vary with the operations involved in the flow entries and flow rules. The effectiveness and efficiency of the proposed approach is examined through HSA.

5.4. Security augmentation

This particular area addresses the promising SDNs that can be extended to enhance security for different applicable networks (Ding et al., 2014). The SDN control plane ensured that sophisticated network-wide policies can be deployed and implemented through simple programs. SDNs can be used to enable operators to implement network policies, such as performing deep packet inspection (Kreutz et al., 2015). Middleboxes can also be used to implement network policies (Sekar et al., 2012; Joseph et al., 2008). Recent studies have proposed the integration of security middleboxes into SDNs to ensure that the programmability aspect is beneficial for security purposes. Two notable contributions and extensions to enhance security by using the distinguishing features of SDNs are Slick (Anwer et al., 2013) and FlowTags architecture (Fayazbakhsh et al., 2013). Slick is a prototypical control plane that allows middleboxes to assist network operators and deploy refined policies efficiently. Moreover, Slick architecture proposes a centralized controller that installs and migrates functions onto middleboxes. Subsequently, the applications can request the corresponding functions to route particular flows based on their prerequisite security criteria. FlowTags architecture is an extension of the SDN architecture and uses flow tracking capability to ensure consistent policy enforcement. The tags are added by middleboxes, which are subsequently used by switches to enforce policies systematically. The FlowTags architecture uses FlowTags APIs to communicate with the controller. FlowTags comprises flow information embedded in a packet header to track and enable supervised routing of the tagged packets. However, the Slick and FlowTags architectures have a serious limitation: they work based mainly on pre-defined policies and cannot handle dynamic operations. In addition, directing traffic through the desired sequence in traditional networks is difficult and requires both operator expertise and manual efforts. Another major contribution is SIMPLE (Qazi et al., 2013), which is a policy enforcement layer and an introduction to an appropriate middlebox deployment that is managed thoroughly by the SDNs. SIMPLE facilitates programming of the entire network by directing a selected traffic through the appropriate middlebox. Realizing the benefits of SDNs for controlling the middlebox-specific traffic steering is the primary goal of the present research proposal. Furthermore, the feasibility of using SDNs along with industry concerns for integration with the existing infrastructure is addressed.

A key contribution is OrchSec (Zaalouk et al., 2014), which is an orchestrator-based architecture that enhances network security to develop reliable security applications. The architecture is based on the use of the SDN control function features and the network monitoring aspects. The proposed architecture decouples the application development process from the SDN controller, thereby making it controller agnostic. The proposed solution can better withstand DoS/DDoS and cache poisoning/ARP spoofing. Researchers (Liu et al., 2011) proposed an SDN-based architecture to restrict covert channel attacks in traditional networks. Tracing the information flow from a low-level host to a high-level host or the same level host, and vice versa, is difficult. The technique employs an OF-based module called “filter,” which is responsible for controlling and checking the packet flow, the content of the packet, and the delay it experiences to avoid side-channel attacks.

The solutions based on the SDN frameworks are as follows: Primarily, an active and live IP address to be attacked should be found. Changing an IP address frequently and proactively is a novel defense mechanism called the moving target defense (MTD). A prime contribution (Jafarian et al., 2012) proposed an SDN-based MTD architecture, where the controller frequently assigns a random virtual IP address to each host translated from the real IP address of the host. Moreover, IP mutation to the end hosts is transparent; however, real IP addresses can only be reached by an authorized access. The proposed mechanism is based on OpenFlow Random Host Mutation (OFRHM), which is used to manage a pool of IP addresses assigned to hosts in a particular network in which the actual IP addresses are hidden from outside the network; this approach represents a specific form of adaptive cybersecurity. The defense against stealthy scanning to discover vulnerable targets in the entire network was presented in Jafarian et al. (2012). This work was inspired by previous research on proactive cyber defense (Atighetchi et al., 2003; Kewley et al., 2001; Antonatos et al., 2007).

Table 5 shows the main classification together with the effect of each of the state-of-the-art security mechanism on SDN layers/interface and its potential.

Although it is a well-understood security problem, DDoS attack detection is still a serious concern. A major contribution toward DDoS detection was presented in Braga et al. (2010). The technique utilizes self-organizing maps (SOMs) to identify abnormal/injected flows. The proposal was based on the SDN programmatic interface, which facilitates switch information handling. The DDoS attack detection is grounded on the flow features. A complementary work that uses SOM is presented in Ramadas et al. (2003), Min and Dongliang (2009), Jiang et al. (2009), Mitrokotsa and Douligeris (2005), Wang et al. (2009). One unique feature is lightweight detection, which can be upgraded easily against new attacks. Moreover, the technique facilitates the addition or removal of switches from the detection loop.

Security monitoring of the entire network is a tedious task; one notable contribution of SDN-enabled security monitoring extension is OpenSAFE (Ballard et al., 2010), which monitors large-scale networks. OpenSAFE is used to manage traffic routing through network monitoring agents by using its ALARMS policy language. ALARMS is a flow specification language that simplifies dramatically the management of the tools used for network monitoring. A similar prominent solution for security monitoring solution in the cloud using SDN features is CloudWatcher (Shin and Gu, 2012), which is a practical and feasible framework for cloud environments and provides security monitoring to supervise the dynamic flows of the network smoothly and efficiently.

SDNs can be useful for network-wide access control. A major contribution is resonance (Nayak et al., 2009), which uses programmable switches and controllers to enable distributed network monitoring by using dynamic access control enforced by network devices themselves. Resonance is a dynamic access control system based on real-time alerts and flow level information; however, it does not follow the centralized architecture. Therefore, resonance is an interface-based policy control. Another recent major contribution for access control is FlowNAC (Jon Matias et al., 2014). FlowNAC is used to secure access to all available network resources and devices. However, its design principle is subjected to a centralized policy decision, and policy enforcement should be defined once to avoid collision. Another proposed solution is mobile application personal policy enforcement router (MAPPER) (Sapio), a fine-grained access control that guarantees network and information security. MAPPER is responsible for imposing user/role specific policies without obtaining access to the device of an end user. The framework can identify and differentiate traffic generated by diverse applications, platforms, and devices. The ident++ protocol solution, which depends on the distributed architecture, is presented in Naous et al. (2009). The proposed protocol queries end users and hosts for information to ensure

Table 5
Classification and identification of the impact of the state-of-the-art security solutions on different SDN layers/interfaces.

Techniques classification	***Techniques***	***SDN layer/interface***				
		Application layer	***North-bound interface***	***Control layer***	***South-bound interface***	***Data layer***
Secure design	FRESCO (Shin et al., 2013)	✓	✓	✓	✓	
	FortNOX (Porras et al., 2012)		✓	✓	✓	
Security audi	VERIFICARE (Skowyra et al., 2013)		✓	✓	✓	
	SDN-Debugger (ndb) (Handigol et al., 2012)				✓	
	Fleet (Matsumoto et al., 2014)			✓	✓	✓
	VAVE (Yao et al., 2011)			✓	✓	✓
Security enforcement policy	FLOVER (Son et al., 2013)	✓	✓	✓	✓	
	Flow-based Security Language (FSL) (Hinrichs et al., 2008)	✓	✓	✓	✓	
	Splendid Isolation: language based security (Schlesinger et al., 2012)		✓	✓		
	NICE (No bugs In Controller Execution) (Canini et al., 2012)	✓	✓		✓	
	LiveSec (Wang et al., 2012)	✓	✓	✓	✓	✓
	SANE (Casado et al., 2006)			✓	✓	✓
	Ethane (Casado et al., 2007)			✓	✓	✓
	PermOF (Wen et al., 2013)	✓		✓	✓	✓
	FLOWGUARD (Hu et al., 2014)	✓	✓	✓	✓	✓
	SDNs firewall using Header Space Analysis (HAS) (Wang et al., 2013)	✓	✓	✓	✓	✓
	VeriFlow (Khurshid et al., 2012)	✓	✓	✓	✓	
	NetPlumber (Kazemian et al., 2013)	✓	✓	✓	✓	
	FlowChecker (Al-Shaer and Al-Haj, 2010)	✓	✓	✓	✓	
Security augmentation	Slick Architecture (Anwer et al., 2013)	✓	✓	✓	✓	✓
	FlowTags Architecture (Fayazbakhsh et al., 2013)	✓	✓	✓	✓	✓
	OrchSec (Zaalouk et al., 2014)	✓	✓	✓		
	Covert channel protection using Filter (Liu et al., 2011)		✓	✓	✓	✓
	SIMPLE (Qazi et al., 2013)	✓		✓		✓
	OF-RHM (Jafarian et al., 2012)			✓	✓	✓
	Self-Organizing Maps (SOM) (Braga et al., 2010)	✓		✓	✓	
	OpenSAFE (Ballard et al., 2010)	✓	✓	✓	✓	
	ident++ (Naous et al., 2009)	✓	✓	✓	✓	✓
	Resonance (Nayak et al., 2009)	✓		✓	✓	✓
	FlowNAC (Jon Matias et al., 2014)	✓		✓	✓	✓
	MAPPER (Sapio)	✓		✓	✓	
	NetFPGA based solution (Goodney et al., 2010)	✓				✓
	ADS Revisit (Mehdi et al., 2011)	✓		✓	✓	
	CloudWatcher (Shin and Gu, 2012)	✓	✓	✓	✓	
	L-IDS (Skowyra et al., 2013)	✓		✓	✓	✓
	Network Intrusion detection and Countermeasure sElection (NICE) (Chung et al., 2013)	✓		✓	✓	✓
	SnortFlow (Xing et al., 2013)	✓		✓		✓
Security monitoring and analysis	OpenWatch (Zhang, 2013)			✓	✓	✓
	FleXam (Shirali-Shahreza and Ganjali, 2013)	✓		✓	✓	
	AVANT-GUARD (Shin et al., 2013)			✓	✓	✓
	NetFuse (Wang et al., 2013)	✓	✓	✓	✓	✓
	CONA (Choi)	✓	✓	✓	✓	✓
Fault tolerance	SPARC (Sharma et al., 2011)			✓	✓	✓
	Link failure (Staessens et al., 2011)			✓	✓	✓

their active involvement in forwarding decisions and to simultaneously avoid bottleneck in the controller.

The SDNs can be better extended in the field of intrusion detection and prevention. An SDN-based learning intrusion detection system (L-IDS) (Skowyra et al., 2013) is used to protect embedded mobile devices in a particular location. The L-IDS detects a wide variety of attacks while reconfiguring the network in real time. The technique helps mitigate the attacks; however, this solution primarily involves delegating network security. A hardware-based solution for network IDS/IPS is presented in Goodney et al. (2010). This technique uses NetFPGA, an open-source field programmable gate array (FPGA) based network interface with OpenFlow modules. The technique is a co-design of hardware/software for developing a deep packet inspection engine. A major contribution in revisiting anomaly-based intrusion detection using SDNs is proposed in Mehdi et al. (2011). The proposed methodology provides high detection accuracy and is suitable small home/small office or home networks. Researchers proved that SDNs are effective for some prominent anomaly detection system algorithms. A significant contribution is NICE in virtual network systems (Chung et al., 2013). NICE is an OF-based framework for the detection of vulnerable applications installed on virtual machines. Subsequently, these vulnerable applications can either be used to compromise VMs in the cloud, especially in infrastructure-as-a-service clouds, or may work as a zombie for further exploitation. NICE is a distributed vulnerability detection framework based on analytical models, programmable virtual switches (OF), and OF programming APIs to build an efficiently monitored control plane to detect and mitigate sophisticated attacks. Another efficient attempt is SnortFlow (Xing et al., 2013), which is an IPS for the cloud environment (Alam et al., 2015). The technique simply combines the capabilities and features of Snort and OF to employ IPS.

NICE and SnortFlow are implemented for the cloud environment with almost the same objectives. However, NICE is an attack graph-based IPS, whereas SnortFlow utilizes snort for intrusion detection and OF-based reconfigurable network features to prevent intrusion.

Remarks: A critical analysis of the literature on SDN security clearly shows that the security research community has two opinions about SDNs. One school of thought is focused on securing SDNs and making them dependable. By contrast, the second school of thought believes in the use of the remarkable features and capabilities of SDNs to enhance and improve the security of different applicable networks; the latter belongs to our main classification of SDN security augmentation. Security augmentation in our classification implies that SDNs are equivalent to "security defined networks." The SDNs contribute greatly in improving the security of many modern and existing cellular, mobile, and wired networks. Although SDNs have promising architecture, their security, reliability, and dependability has not been proven.

5.5. *Security monitoring and analysis*

Security monitoring and analysis is an essential part of the dynamic environment of the SDNs. The authors in Shin et al. (2013) proposed AVANT-GUARD, an implementation of two significant changes to SDNs. One extension to the data plane is called connection migration, which significantly minimizes the data-to-control plane interactions that increase during DoS attacks on a southbound interface. Another extension called actuating trigger expedites the responsiveness to the changing flow dynamics within the SDN data plane. Actuating triggers are introduced over the statistics collection services of the data plane. The proposed model is resilient against different security threats. However, AVANT-GUARD has several limitations. Although AVANT-GUARD works well on network scanning and on transport control protocol (TCP) SYN flooding attacks, it has no counter user datagram protocol, Internet control message protocol, or application layer DoS attacks.

OpenWatch (Zhang, 2013) is an adaptive flow counting method that detects anomalies in the SDNs. Monitoring the network adds an overhead to the network; hence, the proposed model uses an adaptive flow-based counting mechanism to ensure accuracy while anomalies are detected, thereby considerably reducing the overhead. The studies orthogonal to OpenWatch are Yu et al. (2013), Jose et al. (2011), Huici et al. (2012), and Moshref et al. (2013). Unique to the method is its provision of input to the anomaly detectors, simultaneously considering the optimization of the network.

Another contribution is the FleXam (Shirali-Shahreza and Ganjali, 2013), a sampling extension that provides access to an OF controller to obtain packet-level information. The FleXam enables the controller to sample the packets stochastically or deterministically considering the application requirements. FleXam eliminates flow setup time and reduces the control plane load. Consequently, such applications can directly run on small networks.

The authors in Choi recently proposed a content-oriented networking architecture (CONA) that introduced the content-centric communication model to resolve the issue of accountability (Choi). CONA is a content-aware monitoring and supervision method that can be used to detect resource-exhaustive attacks. Moreover, CONA is useful in the detection of a particular malicious host that generates a DDoS attack. Content extraction on the agent helps CONA in attaining accountability and simultaneously taking countermeasures against different DoS attacks. CONA uses NetFPGA and OF-enabled switches.

Another major contribution in this particular area is NetFuse (Wang et al., 2013), which is responsible for monitoring surges due to routing configuration errors, security breaches and attacks, and other operator errors. NetFuse is based on multi-dimensional flow aggregation and detects suspicious flow clusters in cloud and datacenter networks. NetFuse also addresses traffic overloading, which has several serious financial and availability implications in cloud and data center environments.

5.6. *Fault tolerance*

Fault tolerance against different attacks is a serious concern in the SDN dynamic environment, particularly in SDN centralized architecture. Few contributions have been made in this particular area. Researchers in Staessens et al. (2011) conducted different experiments of recovery from a link failure using OF. The technique uses OF-based switches to handle data-to-control plane failures. However, dependency on a centralized architecture may cause delay in restoration and recovery from failure in case of large-scale networks. Split architecture carrier grade networks (SPARK) are another major contribution to this field (Sharma et al., 2011) and ensure fast failure recovery using OF. The SPARK project uses OF-based switches and controllers. Furthermore, this technique is used for fast data-to-control plane recovery failures. The SPARK project aims to develop a fast restoration and recovery mechanism in case of failure by any means. Another solution called automated protection switching (APS) is even faster than SPARK and may not require contacting the controller after failure. Both techniques are highly remarkable contributions; however, the technique in Staessens et al. (2011) caused some delay in large-scale networks.

We briefly illustrate the two main schools of thought followed by a thematic taxonomy.

5.7. *Taxonomy of SDNs security*

The taxonomy shown in Fig. 4 is based on state-of-the-art security solutions for SDN security. The devised taxonomy will help the researchers to understand the problem clearly and to consider all significant aspects of the emerging SDNs.

The existing solutions can be further classified based on the following parameters: solution categories, SDN layers/interfaces, security measures, implementation (simulation/emulation) environment, and security objectives. Researchers contributed toward auditing the dynamic environment of SDNs for security and accountability purposes. Major contributions were based on security policy enforcement and security enhancement of different applicable networks. Several studies focused on security monitoring and analysis, whereas other researchers contributed to fault tolerance.

All these innovative security solutions mainly target issues of a particular layer/interface of the entire SDN architecture. Addressing the issues of different layers/interfaces against various corresponding possible attacks and threats makes these solutions unique. However, these state-of-the-art solutions broadly consider the distinguishing features of SDNs, including centralized management, programmable aspects, flow-based forwarding, and flow analysis, as explained in the introduction. Each security solution of SDNs renders a particular eminent feature that makes these solutions even more exceptional. Moreover, the taxonomy is based on the implementation (simulation/emulation) environment. The majority of the security solutions are based on OF standards and popular OF controllers, as shown in Table 1. The extant security SDN solutions are based purely on a particular security objective.

The taxonomy presents the major security objectives, including policy/rule conflict resolution, rapid designing and development of secure applications, monitoring for security purposes, and malware protection to prevent stealthy scanning and propagation. Moreover, intrusion detection and prevention secure the architecture and defend against different DOS attacks. Auditing and accountability are considered primary objectives for the dynamic environment of SDNs. Furthermore, a key objective is fault tolerance, with focus on the importance of the centralized management architecture of the SDNs.

6. Requirements and key enablers for SDNs security

Ensuring the security of every single component of the SDN is mandatory to build a secure SDN environment. The following are several essential requirements for securing the SDN key components. Figure 5 depicts a situation in which an attacker searches for potential components of SDNs to be compromised. Moreover, 1–4 in Fig. 5 represent the requirements and their key enablers.

6.1. Securing the SDN controller

Securing the SDN controller, which is the central decision point, is the foremost priority. The SDN controller is responsible for the overall management of the network. A simple compromise of the centralized SDN controller can affect the entire network (Metzler, 2012). Furthermore, as a single point of failure, the SDN controller serves as a potential target for attackers. The SDN controller as a software platform, if compromised, essentially allows a hacker to reconfigure the entire network. By spoofing the address of the controller, the attacker can take over the entire network easily through a fake controller.

This key component needs to be protected, and it can be protected in the following ways:

- High availability of the controller is ensured to protect against different DoS and DDoS attacks.
- The SDN controller must be protected with security policy enforcement, high availability, and minimum possible delay during incoming packets (Vissicchio et al., 2014).
- The OS that contains the controller must be secured against exploitable patches, backdoor accounts, and open doors, such as vulnerable open ports, services, and protocols.

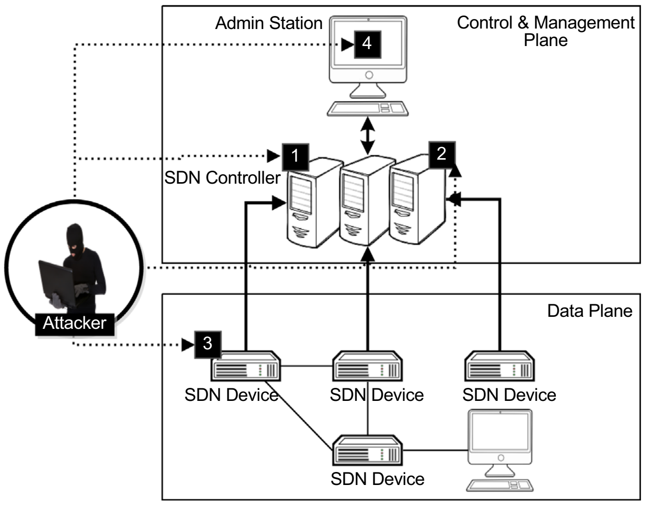

Fig. 5. Attacker searching for potential targets.

- The protection of the system with the SDN controller must be secured against physical threats.
- The controller must have a mechanism that alerts an administrator to limit control communication during the attack or in case of a sudden attack (Jammal et al., 2014).
- An intelligent access control list must be implemented for packet filtration, and full isolation among the tenants that share the infrastructure must be ensured (Jammal et al., 2014).

6.2. Protecting flow paradigm of the SDN

SDN grounded on flow-based forwarding can ensure end-to-end communication security. The flow paradigm is the soul of SDN and must be protected. A successful influx of bogus flow may compromise the entire network (Bing et al., 2014). Flow abstraction of the controller may result in harvesting of the intelligence of the connected resources, and such harvested intelligence can be used in further attacks and exploitation (Kreutz et al., 2015; Bing et al., 2014). An updated access control mechanism should be deployed in the network. Moreover, flows should be encrypted to prevent the injection of malicious flows. Proper authentication and authorization should be implemented to prevent side-channel attacks.

6.3. Fortifying SDN agents

The security of the SDN agent is important because it constitutes the data plane environment To compromise a strong entity, such as the SDN controller, an attacker may reach the target by compromising any vulnerable agent of SDN. For instance, link layer discovery protocol packets with forged source addresses can cause the SDN controller to install flow rules grounded on bogus information. Moreover, many existing switches, as part of the SDN infrastructure layer, are by default in listener mode, which may easily lead to the launch of malicious connections (Kreutz et al., 2013; Costa). Injecting false flow at any SDN agent can lead to its distribution to numerous agents who ultimately cause serious network disturbance. The security of the SDN agents requires deploying the latest identity management, threat isolation, and mitigation techniques. Moreover, the SDN agents require physical security. Further IPS, IDS, and firewalls should be actively deployed.

6.4. Hardening application programming interfaces (APIs) and communication channels

APIs can be a potential target for attackers (Kreutz et al., 2015; Scott-Hayward et al., 2013). Most importantly, the southbound APIs can be targeted easily for different DoS attacks to make the entire network unavailable. The creation of malicious APIs by skilled programmers is a critical issue; this trend already exists in the security research community. The communication channel between each layer must be well protected (Scott-Hayward et al., 2013); security measures include secure coding, deployment of integrity checks, and digital signing of the code. Moreover, the communication channels can be hardened by using TLS/SSL security or other cryptographic alternatives, such as threshold cryptography (Kreutz et al., 2013; Sookhak et al., 2015; Sookhak et al., 2014).

7. Open issues for securing SDNs

Security plays a vital role in deploying SDNs across different applicable networks. SDNs are receiving attention because of their diverse applicability (Jarraya et al., 2014; Ding et al., 2014; Baldini et al., 2012; Yoo, 2014; Pfeiffenberger and Jia Lei, 2014; Yang et al., 2014). However, security is one of the key obstacles that hinder the growth and overall adoption of SDNs. This section elaborates open security issues, challenges, and foreseeable directions that might facilitate the wide acceptance of SDNs. Figure 6 presents the future security challenges, trends, and direction of SDNs.

7.1. Security and SDNs virtualization

The use of SDNs in network virtualization has introduced many security issues that need to be addressed (Hegde and Hu, 2014; Bays et al., 2015; You et al., 2014). Several unaddressed issues related to SDNs deployed for network virtualization are presented. These issues were identified when the OF protocol was used for network virtualization.

FlowVisor (Sherwood et al., 2009), a special-purpose controller, is used to ensure isolation between multiple virtual networks. In addition, it acts as a transparent proxy between a controller and a switch. FlowVisor configures the entire network into different slices. Moreover, it facilitates rewriting control messages according to user-defined policies to guarantee isolation. Furthermore, FlowVisor acts as a slicer and a controller in an SDN environment. Subsequently, it becomes a potential target for attackers because the entire network is compromised once it is down.

The issue of confidentiality, integrity, and availability to ensure network security must be addressed to achieve secure and dependable SDNs (Romão et al., 2013). FlowVisor does not implement action isolation (Costa); consequently, a controller can set any type of action on a flow entry without any control of the FlowVisor.

Victor (Costa) proposed three possible threats: (a) VLAN ID access problem: This problem occurs when a controller is denied access to any VLAN ID, whereas FlowVisor is allowed to create flow entries whose action can change the VLAN ID of a packet. This situation creates an opportunity for a nasty controller to inject packets into another slice or can be utilized to steal packets; (b) Field rewrite problem: This problem occurs when a controller is given access to a particular VLAN ID tag to create a flow entry whose action can change the VLAN ID of its own packet, thereby creating an opportunity for a malicious controller to inject packets into another slice or to steal packets. Essentially, in a virtual environment, the header fields involved in creating a slice can experience the same problem. It can be a modification of any header field, such as IP/MAC source/destination address and transport source/destination numbers; (c) Wildcard rewrite problem: This problem occurs when a controller is given access to a transport source port A, and the controller wants to create a flow entry with an unspecified transport source port (wildcard). FlowVisor should rewrite the valid transport source (A) to the wildcard value; however, in reality, this rewriting valid transport source does not occur, and it simply rewrites the wildcard value with any matching transport source port and may raise serious security concerns. The problem is the same for other fields, such as protocol type and transport destination. Malicious injections may occur by simply following the existing field rewrite problem, which allows the end user to change the VLAN ID tag in particular circumstances. Table 6 summarizes the security problems of the SDN virtual environment.

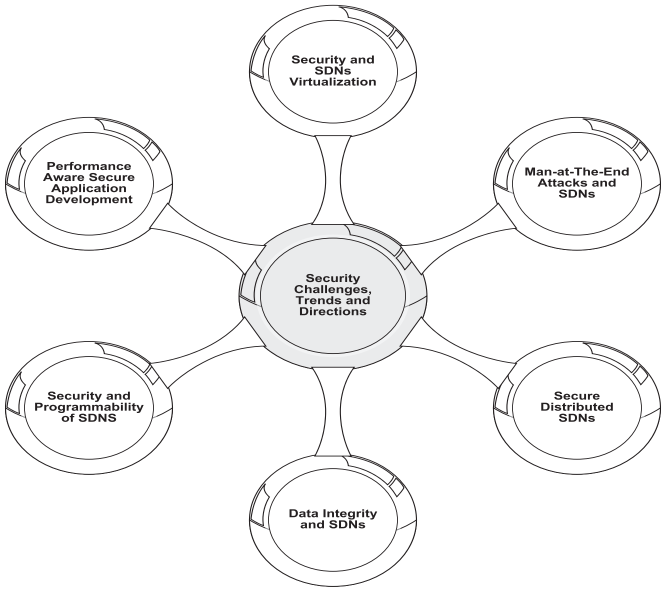
Security and SDNs Virtualization
Performance Aware Secure Application Development
Man-at-The-End Attacks and SDNs
Security Challenges, Trends and Directions
Security and Programmability of SDNS
Secure Distributed SDNs
Data Integrity and SDNs

Fig. 6. The SDNs security challenges, trends and directions.

7.2. SDN controller-specific security issues in virtual environments

The controller remains a potential target for attackers and is the most likely target of the first line of attack. The following are several unaddressed scenarios when the controller is targeted in a virtual environment (Hegde and Hu, 2014; Bays et al., 2015). Figure 5 illustrates the particular scenario of using OpenVirteX, a special controller used to create virtual networks. When using OpenVirteX, several controllers, such as POX (POX, 2014) controller, are placed on the end user side, as shown in Fig. 5. Although POX has many advantages, it is also exploitable. Using OpenVirteX instead of FlowVisor addresses space isolation. However, it does not implement action isolation, that is, a controller can set any type of action on a flow entry without any control by OpenVirteX (Al-Shabibi et al., 2014). Figure 7 shows the following attacks:

i) Denial of service (DoS) attack: A POX (POX, 2014) controller is placed on the end user side to create a particular virtual network using OpenVirteX. POX controller possesses critical knowledge of the network and is prone to many attacks, particularly DoS attacks (Bing et al., 2014). An attacker can generate a large number of flows to bring the network down or make it work improperly by targeting OpenVirteX.
ii) Spoofing attack: The spoofing attack can be illustrated considering the same scenario discussed above. For example, the floodlight controller is already aware of the IP address of OpenVirteX and can merely forge the IP address of OpenVirteX to launch a simple and easy spoofing attack.
iii) Malicious injection: Malicious injections may be performed by using the existing field rewrite problem, which allows the end user to change the VLAN ID tag in particular circumstances (Bing et al., 2014). Malicious injection creates an opportunity for a nasty controller to inject packets into another slice. OpenVirteX does not implement action isolation, that is, a controller can set any type of action in the flow entry without any control of the controller in this particular case.

7.3. Man-at-the-end attacks and SDNs

Any type of SDN, regardless of its architecture, design, configuration, and maintenance, relies on people. Traditional computer and network security for home networks, the Internet, the cloud, SDNs, and the Internet of Things are inadequate to address MATE attacks (Akhunzada et al., 2015). MATE attacks are fundamentally difficult to resolve under general circumstances (Falcarin et al., 2011; Gu et al., 2011; Collberg et al., 2011; Ceccato et al., 2013). The problematic part is that humans have become the edge, and auditing the human mind is a complex task. Moreover, we cannot

Table 6
Security problems of SDNs virtual environment

SDNs security problems in virtual environment	*Malicious injection*	*Stealing packets*	*Denial of service (DoS)*	*Spoofing attack*	*Status*
VLAN ID access problem	✓	✓			Unexplored
Field rewrite problem	✓				Unexplored
Wildcard rewrite problem		✓			Unexplored
Implementation of action isolation					Unexplored
Denial of service (DoS)			✓		Unexplored
Spoofing attack				✓	Unexplored

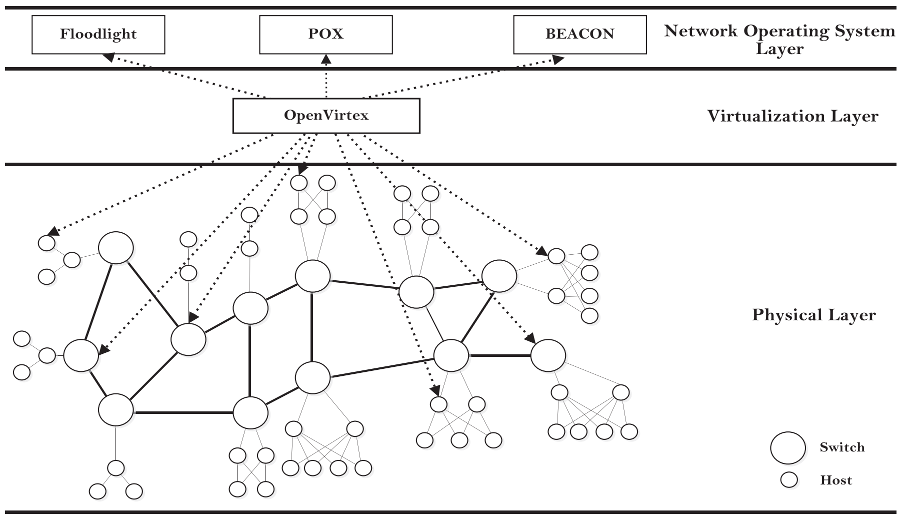

Fig. 7. Controller-specific attack scenarios of creating virtual environment using OpenVirtex.

install antivirus on a system administrator's cerebral cortex (Akhunzada et al., 2015). We still rely on perimeter defense even though the reality is that in digital world boundaries are non-existent. We continue to imagine that we have control over devices when we do not. We expect users to adhere to policies; however, we have no control over policy enforcement. SDNs may be targeted by MATE attacks in many ways. MATE attackers can primarily target the centralized control management architecture of SDNs, which is a single point of failure that may lead to the breakdown of a network. Furthermore, MATE attackers may penetrate and bypass the programmable aspects of SDNs, as these attackers are highly skilled and can compromise any SDN agent for further exploitation. A serious MATE problem was recently addressed in Matsumoto et al. (2014). The present paper can provide ideas for maintaining data secrecy if a network is under malicious control. A detailed description of possible MATE attack scenarios is beyond the scope of this study. MATE attacks need to be addressed to ensure secure and dependable SDNs. Figure 8 depicts a situation in which a MATE attacker has full access to his or her capabilities to enable him or her to bypass SDN protection mechanisms.

7.4. *Performance aware secure applications development*

The SDN centralized architecture does not bear long latencies unlike traditional networks. The northbound API of the SDN facilitates the development of a set of desired security applications at the application layer of the SDN. These security applications need to access packet-level information at different levels to be effective, particularly for digital forensics, intrusion detection, and prevention (Khan et al., 2014). Moreover, accessing the payload is obligatory where DPI is required (Antonello et al., 2012). Subsequently, obtaining the required information has considerable latency, which causes the entire traffic to behave abnormally. Furthermore, in a special case of the first packet with unknown flow and having no buffer availability in the switch, the current OF version (Specification-Version, 2013) may send the entire packet to the SDN controller. Thus, a security application that manipulates deep packet inspection cannot benefit from the current version of OF in this particular case, thereby degrading the overall performance. Some studies (Shin et al., 2013; Giotis et al., 2014; Shirali-Shahreza and Ganjali., 2013) were conducted to address this issue. Serious effort should be made to develop performance-aware security applications. Studies should also focus on developing security applications with good trade-offs among security, performance, and usability.

7.5. *Secure and dependable SDNs*

SDN security and dependability are open issues (Kreutz et al., 2015; Jammal et al., 2014; Hakiri et al., 2014; Vissicchio et al., 2014). Several studies on security have focused on enhancing the security of different networks by using SDNs. SDNs have a promising architecture, yet they might pose potential security risks and possible threats to a network. Literature on SDNs indicates that security and dependability have not yet been achieved (Kreutz et al., 2013).

Several potential threat vectors and vulnerabilities during the deployment of SDNs using OF were investigated (Kreutz et al., 2013; Klöti, 2013). Unlike traditional network security attacks, SDNs introduce new threat vectors (Kreutz et al., 2013), particularly those related to the SDN controller and the southbound and the northbound communication interfaces. These three threat vectors, which are identified in Kreutz et al. (2013), are

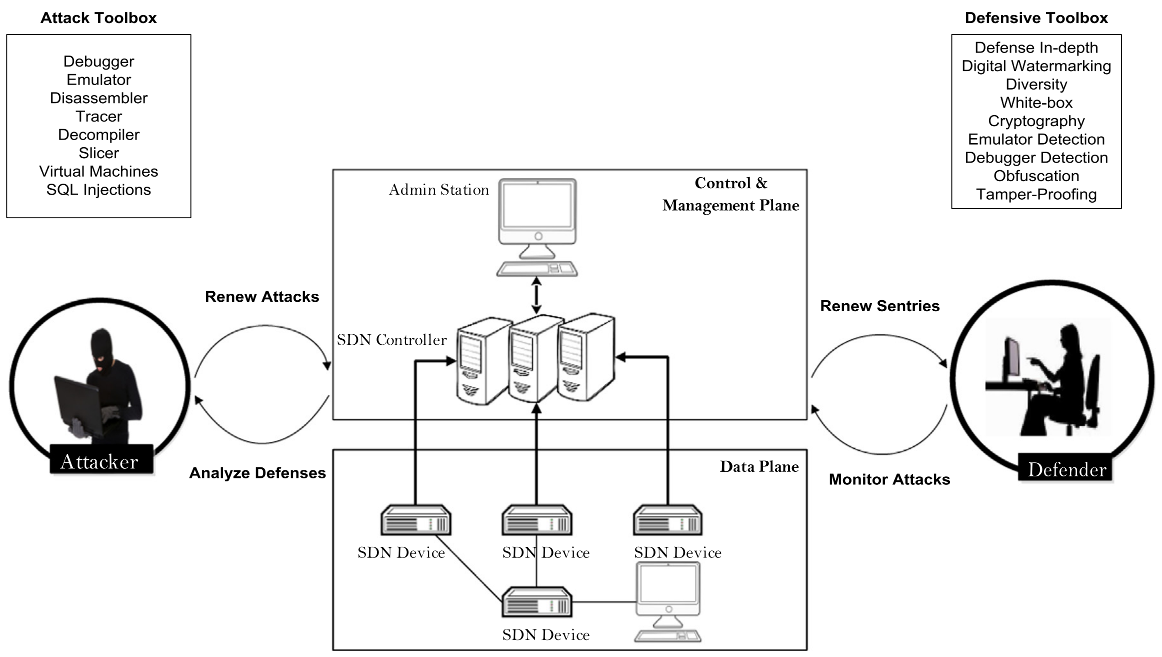

Fig. 8. Depicts that a man-at-the-end attacker has full access to analyze and utilize his or her capabilities to bypass the SDNs protections.

predominantly related to SDNs, whereas other threat vectors are common. However, these threat vectors may have a more devastating and augmented effect on SDNs than on traditional networks.

7.6. *Programmability and SDNs*

The SDN has to cope with the potential set of complex problems (Wasserman and Hartman, 2013) more related to the programmability aspects of the network. The tendency of launching sophisticated DDoS, phishing, spam and malware attacks is expected to increase massively. Subsequently, it will change the dynamics surrounding the infrastructures of secure SDNs. Moreover, if we see SDNs in the context of mobile wireless networks (Yang et al., 2014; Leung et al., 2014; Woon Hau et al., 2014), then the possibility of injection and eavesdropping (active, passive) is much higher because of broadcast and extant vulnerabilities of the wireless channels. Furthermore, the case of mobile ad hoc networks where typical security solutions are entirely infeasible to implement owing to their lack of infrastructure (e.g., security servers) becomes increasingly serious and complex (Hakiri et al., 2014). Classical security solutions necessitate downtime to orchestrate topological changes during reconfiguration, and configuration, when security services are debugged and turned on.

7.7. *Data integrity and SDNs*

Security was not considered as part of the initial SDN design; thus, the certificate format to ensure data integrity is not well described by Open-Flow(OF) specifications (Hakiri et al., 2014; Benton et al., 2013). Sophisticated authentication and encryption mechanisms are needed to recover from packet failure and to prevent hackers from violating data integrity. However, the current OF specification recommends the use of TLS, which provides encrypted secure channels (Lara et al., 2014). Subsequently, eavesdropping is prevented. Yet TLS is unreliable (Kreutz et al., 2013) in many cases. The OF specifications provide no details on the use of interoperable versions. Moreover, establishing a secure connection and checking the certificates between the switches or the controllers are not specified. For example, mutual authentication among multiple controllers is performed through an exchange of certificates signed by a private key of a third party. The difference between these keys results in serious security vulnerabilities, and the controllers may not be able to interoperate (Hakiri et al., 2014; Kloti et al., 2013). To have more sophisticated cryptographic alternatives, this particular area needs to be well addressed.

7.8. *Distributed SDNs and security*

SDNs introduce promising monitoring and management abilities in small to medium-sized network topologies. However, security compliance and policy enforcement, troubleshooting, debugging, and monitoring are problematic in distributed SDNs (Hakiri et al., 2014). Moreover, the management decisions of SDNs are based on the behavior and information of the network. Furthermore, the OF SDN session is subject to different actions specified in the flow table (Lara et al., 2014; Banjar et al., 2014). Responding to a wide range of diverse events, such as intrusions, necessitates the enforcement of high-level policies by corresponding network operators. However, SDNs offer little to no automatic response mechanism for such events (Hakiri et al., 2014). Event-driven programming interfaces for service providers are needed to manage diverse asynchronous events linked with the corresponding network. The configuration of a carrier-grade network in particular remains a tedious task, given that administrators have to cope with vendor-specific and low-level interfaces to implement high-level functions. Consequently, low-level configurations remain a difficult task in distributed SDNs. These issues in relation to distributed SDNs need to be closely addressed.

8. Conclusion

The emergence of SDNs has resulted in additional security requirements because of newly deployed infrastructural entities. Despite the promising architecture of SDNs, security was not considered part of the initial design. Significant work is in progress to develop state-of-the-art SDN security applications and solutions. However, research on SDN security is still in its infancy. Secure and dependable SDNs remains a distant goal.

To meet the newly imposed network security requirements, this study presented a broad overview of the security implications of each SDN layer/interface. We devised a contemporary layered/interface taxonomy of the reported security vulnerabilities, attacks, and challenges of SDNs to illustrate the main categories of security implications for each SDN layer/interface. Furthermore, we highlighted and analyzed the possible threats that may affect and target a particular layer/interface with a suggested corresponding compact solution. A discussion on state-of-the-art security solutions was also presented. A comprehensive survey, analysis, and classification of extant SDN security solutions promote ways to secure dependable SDNs. The surveyed solutions were further classified by devising a thematic taxonomy based on the SDN layers/interfaces, the SDNs' eminent features, implementation environment, and security objectives. This study identified the SDNs' distinguishing features for each state-of-the-art security solution, thereby making these security solutions unique. Moreover, the potential effects of each security solution on different SDN layers/interfaces were identified and presented. This study illustrated two main schools of thought in the SDN security research community. The potential security implications with their key enablers were elaborated for the development of secure and dependable SDNs. Finally, we presented open security issues, challenges, and future research directions that may assist in the wide acceptance of the emerging SDNs.

Acknowledgments

This work is fully funded by Bright Spark Unit, University of Malaya, Malaysia and partially funded by Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant UM.C/625/1/HIR/MOE/FCSIT/03.

References

Ali ST, et al. A survey of securing networks using software defined networking. IEEE Trans Reliab 2014.

Akhunzada A, et al. Securing software defined networks: taxonomy, requirements, and open issues. Commun Mag, IEEE 2015;53(4):36–44.

Ahmad RW, et al. A review on mobile application energy profiling: taxonomy, state-of-the-art, and open research issues. J Netw Comput Appl 2015.

Attacks S. ⟨http://sdnsecurity.org/project_overview.html⟩.

Al-Shaer E, Al-Haj S. FlowChecker: configuration analysis and verification of federated OpenFlow infrastructures. In: Proceedings of the 3rd ACM workshop on assurable and usable security configuration; 2010.

Al-Shaer E, et al. Network configuration in a box: towards end-to-end verification of network reachability and security. In: Proceedings of the 17th IEEE international conference on network protocols, ICNP 2009.

Anwer B, et al. A slick control plane for network middleboxes. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking; 2013.

Atighetchi M, et al. Adaptive use of network-centric mechanisms in cyber-defense. In: Sixth IEEE international symposium on object-oriented real-time distributed computing; 2003.

Antonatos S, et al. Defending against hitlist worms using network address space randomization. Comput Netw 2007;51(12):3471–90.

Alam KA, et al. Impact analysis and change propagation in service-oriented enterprises: a systematic review. Inf Syst 2015;54:43–73.

Al-Shabibi A, et al. OpenVirteX: make your virtual SDNs programmable. In: Proceedings of the third ACM workshop on hot topics in software defined networking; 2014.

Akhunzada A, et al. Man-at-the-end attacks: analysis, taxonomy, human aspects, motivation and future directions. J Netw Comput Appl 2015;48(0):44–57.

Antonello R, et al. Deep packet inspection tools and techniques in commodity platforms: challenges and trends. J Netw Comput Appl 2012;**35**(6):1863–78.

Benton K, Camp LJ, Small C. Openflow vulnerability assessment. In: Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking; 2013.

Baldini G, et al. Security aspects in software defined radio and cognitive radio networks: a survey and a way ahead. Commun Surv Tutor, IEEE 2012;14(2):355–79.

Berde P, et al. ONOS: towards an open, distributed SDN OS. In: Proceedings of the third workshop on hot topics in software defined networking; 2014.

Blanchet B. ProVerif automatic cryptographic protocol verifier user manual. CNRS, Departement dInformatique. Paris: Ecole Normale Superieure; 2005.

Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: Proceedings of 2010 IEEE 35th conference on local computer networks (LCN), 2010.

Ballard JR, Rae I, Akella A. Extensible and scalable network monitoring using opensafe. In: Proceedings of INM/WREN; 2010.

Bing W, et al. DDoS attack protection in the era of cloud computing and software-defined networking. In: Proceedings of IEEE 22nd international conference on network protocols (ICNP); 2014 .

Bays LR, et al. Virtual network security: threats, countermeasures, and challenges. J Internet Serv Appl 2015;6(1):1–19.

Banjar A, et al. Analysing the performance of the OpenFlow standard for software-defined networking using the OMNeT network simulator. In: 2014 Asia-Pacific conference on computer aided system engineering (APCASE); 2014.

Cai Z, Cox AL, Ng TE. Maestro: a system for scalable openflow control. Structure 2010.

Canini M, et al. A NICE way to test OpenFlow applications. In: NSDI; 2012.

Casado M, et al. SANE: a protection architecture for enterprise networks. In: Usenix Security; 2006.

Casado M, et al. Ethane: taking control of the enterprise. ACM SIGCOMM Comput Commun Rev 2007;37(4):1–12.

Chung C-J, et al. NICE: network intrusion detection and countermeasure selection in virtual network systems. IEEE Trans Dependable Secur Comput 2013:1.

Choi Y. Implementation of content-oriented networking architecture (CONA): a focus on DDoS countermeasure.

Costa V.T.C.a.L.ı.H.M.K., Vulnerability study of flowvisor-based virtualized network environments.

Collberg C, et al. Toward digital asset protection. Intell Syst, IEEE 2011;**26**(6):8–13.

Ceccato M, et al. A family of experiments to assess the effectiveness and efficiency of source code obfuscation techniques. Empir Softw Eng 2013:1–35.

Ding AY, et al. Software defined networking for security enhancement in wireless mobile networks. Comput Netw 2014;66:94–101.

Doria A, et al., Forwarding and control element separation (ForCES) protocol specification. 2010.

Dover JM, A denial of service attack against the Open Floodlight SDN controller. Dover Networks LCC.

Dhawan M, et al. SPHINX: detecting security attacks in software-defined networks. In: Proceedings of the 2015 network and distributed system security (NDSS) symposium; 2015.

Ding AY, et al. Software defined networking for security enhancement in wireless mobile networks. Comput Netw 2014;66(0):94–101.

Erickson D. Floodlight Java based OpenFlow Controller. Last accessed, Ago, 2012.

Erickson D. The beacon openflow controller. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking; 2013.

El-Atawy A, et al. An automated framework for validating firewall policy enforcement. In: Eighth IEEE international workshop on policies for distributed systems and networks, POLICY'07; 2007.

Fei H, Qi H, Ke B. A survey on software-defined network and OpenFlow: from concept to implementation. Commun Surv Tutor, IEEE 2014;16(4):2181–206.

Farhady H, Lee H, Nakao A. Software-defined networking: a survey. Comput Netw 2015;81(0):79–95.

Fayazbakhsh SK, et al. FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions. In: Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking; 2013.

Falcarin P, et al. Guest editors' introduction: software protection. Software, IEEE 2011;28(2):24–7.

Gude N, et al. NOX: towards an operating system for networks. ACM SIGCOMM Comput Commun Rev 2008;38(3):105–10.

Govindarajan K, Kong Chee M, Hong O. A literature review on software-defined networking (SDN) research topics, challenges and solutions. In: Proceedings of the fifth International conference on advanced computing (ICoAC); 2013.

Garnaev A, Trappe W. To eavesdrop or jam, that is the question. Ad Hoc Networks. Springer; 146–61.

Goodney A, et al. Pattern based packet filtering using NetFPGA in DETER infrastructure. In: 1st Asia NetFPGA developers workshop. Daejeon, Korea; 2010.

Gu YX, Wyseur B, Preneel B. Software-based protection is moving to the mainstream. IEEE Softw 2011;28:2.

Giotis K, et al. Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Comput Netw 2014;62:122–36.

Hakiri A, et al. Software-defined networking: challenges and research opportunities for future internet. Comput Netw 2014;75:453–71.

Hernan S, et al. Threat modeling-uncover security design flaws using the stride approach. MSDN Magazine-Louisville; 68–75.

Hong S, et al. Poisoning network visibility in software-defined networks: new attacks and countermeasures, NDSS; 2015.

Hartman S, Wasserman M, Zhang D. Security requirements in the software defined networking model. IETF Draft (draft-hartman-sdnsec-requirements) 2013.

Handigol N, et al. Where is the debugger for my software-defined network? In: Proceedings of the first ACM workshop on Hot topics in software defined networks; 2012.

Hinrichs T, et al. Expressing and enforcing flow-based network security policies, Technical Report. University of Chicago; 2008.

Hu H, et al. FLOWGUARD: building robust firewalls for software-defined networks. In: Proceedings of the third ACM workshop on Hot topics in software defined networking; 2014.
Huici F, et al. Blockmon: a high-performance composable network traffic measurement system. ACM SIGCOMM Comput Commun Rev 2012;42(4):79–80.
HEGDE N, HU F. seCurity issues in sdn/Open FlOw. Network Innovation through OpenFlow and SDN: Principles and Design 2014:415.
Jarraya Y, Madi T, Debbabi M. A survey and a layered taxonomy of software-defined networking. Commun Surv Tutor, IEEE 2014;16(4):1955–80.
Jammal M, et al. Software defined networking: state of the art and research challenges. Comput Netw 2014;72(0):74–98.
Jain R, Paul S. Network virtualization and software defined networking for cloud computing: a survey. Commun Mag, IEEE 2013;51(11):24–31.
Jafarian JH, Al-Shaer E, Duan Q. Openflow random host mutation: transparent moving target defense using software defined networking. In: Proceedings of the first ACM workshop on Hot topics in software defined networks; 2012..
Jon Matias JG, Alaitz Mendiola, Nerea Toledo, Eduardo Jacob, FlowNAC: flow-based network access control; 2014.
Joseph DA, Tavakoli A, Stoica I. A policy-aware switching layer for data centers. In: ACM SIGCOMM computer communication review; 2008.
Jiang D, Yang Y, Xia M. Research on intrusion detection based on an improved SOM neural network. In: Proceedings of the fifth international conference on information assurance and security, IAS'09; 2009.
Jose L, Yu M, Rexford J. Online measurement of large traffic aggregates on commodity switches. In: Proceedings of the USENIX HotICE workshop; 2011.
Jammal M, et al., Software-defined networking: state of the art and research challenges. arXiv preprint arXiv:1406.0124; 2014.
Kreutz D, et al. Software-defined networking: a comprehensive survey. Proc IEEE 2015;103(1):14–76.
Kloti R, Kotronis V, Smith P. OpenFlow: a security analysis. In: Proceedings of the 21st IEEE international conference on network protocols (ICNP); 2013.
Kreutz D, Ramos F, Verissimo P. Towards secure and dependable software-defined networks. In: Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking; 2013.
Kwiatkowska M, Norman G, Parker. D. PRISM 4.0: verification of probabilistic real-time systems. Computer aided verification. Springer; 2011.
Khurshid A, et al. Veriflow: verifying network-wide invariants in real time. ACM SIGCOMM Comput Commun Rev 2012;42(4):467–72.
Kazemian P, et al. Real time network policy checking using header space analysis. NSDI 2013.
Kothari N, et al. Finding protocol manipulation attacks. In: ACM SIGCOMM computer communication review; 2011.
Koponen T, et al. Onix: a distributed control platform for large-scale production networks. In: OSDI; 2010.
Kewley D, et al. Dynamic approaches to thwart adversary intelligence gathering. In: Proceedings of DARPA information survivability conference & exposition II, DISCEX'01; 2001.
Khan S, et al. A comprehensive review on adaptability of network forensics frameworks for mobile cloud computing. Sci World J 2014;2014.
Klöti R. Openflow: a security analysis. In: Proceedings of IEEE workshop on secure network protocols (NPSec); 2013.
Liu Y, et al. Optimal scheduling for multi-flow update in software-defined networks. J Netw Comput Appl 2015;54(0):11–9.
Lara A, Kolasani A, Ramamurthy B. Network innovation using OpenFlow: a survey. IEEE Commun Surv Tutor 2014;16(1):493–512.
Liyanage M, Gurtov A. Secured VPN models for LTE backhaul networks. In: Proceedings of the vehicular technology conference (VTC Fall); 2012.
Li C, Raghunathan A, Jha NK. An architecture for secure software defined radio. In: Proceedings of the conference on design, automation and test in Europe. France: European Design and Automation Association: Nice; 448–53.
Liu AX. Formal verification of firewall policies. In: IEEE international conference on communications, ICC'08, 2008.
Liu X, et al. Design of the multi-level security network switch system which restricts covert channel. In: Proceedings of the IEEE 3rd international conference on communication software and networks (ICCSN); 2011.
Leung VCM, et al. Unveiling 5G wireless networks: emerging research advances, prospects, and challenges [Guest Editorial]. Netw IEEE 2014;28(6):3–5.
Lara A, Kolasani A, Ramamurthy B. Network innovation using OpenFlow: a survey. Commun Surv Tutor, IEEE 2014;16(1):493–512.
Macedo D, et al. Programmable networks-from software defined radio to software defined networking. Commun Surv Tutor, IEEE 2015(99): 1-1.
McKeown N, et al. OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 2008;38(2):69–74.
Medved J, et al. OpenDaylight: towards a model-driven SDN controller architecture. In: IEEE 15th international symposium on a world of wireless, mobile and multimedia networks (WoWMoM); 2014.
Monsanto C, et al. Composing software defined networks. In: NSDI; 2013.
Mogul JC, et al. Devoflow: cost-effective flow management for high performance enterprise networks. In: Proceedings of the 9th ACM SIGCOMM workshop on hot topics in networks; 2010.
Matsumoto S, Hitz S, Perrig A. Fleet: defending SDNs from malicious administrators. In: Proceedings of the third workshop on hot topics in software defined networking; 2014.
Mehdi SA, Khalid J, Khayam SA. Revisiting traffic anomaly detection using software defined networking. Recent advances in intrusion detection. Springer; 2011.
Min L, Dongliang W. Anormaly intrusion detection based on SOM. In: Proceedings of the WASE international conference on information engineering, ICIE'09; 2009.
Mitrokotsa A, Douligeris C. Detecting denial of service attacks using emergent self-organizing maps. In: Proceedings of the fifth IEEE international symposium on signal processing and information technology; 2005.
Moshref M, Yu M, Govindan R. Resource/accuracy tradeoffs in software-defined measurement. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking; 2013.
Metzler J. Understanding software-defined networks. Reports. information. com, 2012. 9.
Nunes BAA, et al. A survey of software-defined networking: past, present, and future of programmable networks. Commun Surv Tutor, IEEE 2014;16(3):1617–34.
Ng E. Maestro: a system for scalable OpenFlow control.
Naous J, et al. Delegating network security with more information. In: Proceedings of the 1st ACM workshop on research on enterprise networking; 2009.
Nayak AK, et al. Resonance: dynamic access control for enterprise networks. In: Proceedings of the 1st ACM workshop on research on enterprise networking; 2009.
Pfeiffenberger T, Jia Lei D. Evaluation of software-defined networking for power systems. In: Proceedings of IEEE international conference on Intelligent energy and power systems (IEPS); 2014.
POX ⟨http://www.noxrepo.org⟩ [accessed 27.08.14].
Pan P, Nadeau T. Software driven networks problem statement 2011.
Porras P, et al. A security enforcement kernel for OpenFlow networks. In: Proceedings of the first ACM workshop on Hot topics in software defined networks; 2012.
Qazi ZA, et al. Simple-fying middlebox policy enforcement using SDN. In: Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM; 2013.
RouteFlow ⟨http://cpqd.github.io/RouteFlow/⟩ [accessed 11.11.14].
Ryu ⟨http://osrg.github.io/ryu/⟩ [accessed 11.11.14].
Ramadas M, Ostermann S, Tjaden. B. Detecting anomalous network traffic with self-organizing maps. Recent advances in intrusion detection. Springer; 2003.
Romão D, et al. Practical security analysis of OpenFlow implementation; 2013.
Scott-Hayward S, O'Callaghan G, Sezer S. SDN security: a survey. In: EEE SDN for future networks and services (SDN4FNS); 2013.
Shuja J, et al. Survey of techniques and architectures for designing energy-efficient data centers. Syst J, IEEE 2014;99:1–13.
Shimonishi H, et al. Trema: an open source OpenFlow controller platform. GEC-11 Poster; 2011.
Sherwood R, et al. Flowvisor: a network virtualization layer. OpenFlow Switch Consortium, Tech. Rep 2009.
SNAC ⟨http://www.openflowhub.org/display/Snac/SNAC+Home⟩ [accessed 11.11.14].
Shimonishi H, et al. Helios: fully distributed OpenFlow controller platform. In: Proceedings of the 9th GENI engineering conference (GEC9) Demo; 2010.
Shin S, et al. Rosemary: a robust, secure, and high-performance network operating system. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security; 2014.
Shin S, et al. A framework for integrating security services into software-defined networks. In: Proceedings of the 2013 open networking summit (Research Track poster paper), ser. ONS, 2013. 13.
Sezer S, et al. Are we ready for SDN? Implementation challenges for software-defined networks Commun Mag, IEEE 2013;51(7):36–43.
Sakaguchi K, et al. ACU and RSM based radio spectrum management for realization of flexible software defined radio world. IEICE Trans Commun 2003;86 (12):3417–24.
Shin S, et al. Fresco: modular composable security services for software-defined networks. Internet Society NDSS; 2013.
Skowyra RW, et al. Verifiably-safe software-defined networks for CPS. In: Proceedings of the 2nd ACM international conference on high confidence networked systems; 2013.
Saucez D, Bonaventure O, Iannone L. LISP Threats Analysis 2013.
Son, S., et al. Model checking invariant security properties in OpenFlow. in Communications (ICC), 2013 IEEE International Conference on. 2013. IEEE.
Schlesinger, C., et al. Splendid Isolation: Language-Based Security for Software-Defined Networks. in Proc. of Workshop on Hot Topics in Software Defined Networking. 2012.
Sapio, A., et al., MAPPER: a Mobile Application Personal Policy Enforcement Router for Enterprise Networks.
Shin S, Gu G. CloudWatcher: network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?). In: Proceedings of the 20th IEEE international conference on network protocols (ICNP); 2012.
Skowyra R, Bahargam S, Bestavros A. Software-defined IDS for securing embedded mobile devices. In: Proceedings of the IEEE high performance extreme computing conference (HPEC); 2013.
Shirali-Shahreza S, Ganjali Y. FleXam: flexible sampling extension for monitoring and security applications in openflow. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking; 2013.
Shin S, et al. AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the 2013 ACM SIGSAC conference on computer & communications security; 2013.
Sharma S, et al. Enabling fast failure recovery in OpenFlow networks. In: 8th IEEE International workshop on the design of reliable communication networks (DRCN); 2011.

Staessens D, et al. Software defined networking: meeting carrier grade requirements. In: 2011 18th IEEE workshop on local & metropolitan area networks (LANMAN); 2011.

Sekar V, et al. Design and implementation of a consolidated middlebox architecture. In: NSDI; 2012.

Sookhak M, et al. Remote data auditing in cloud computing environments: a survey, taxonomy, and open issues. ACM Comput Surv 2015;47(4):1–34.

Sookhak M, et al. Towards dynamic remote data auditing in computational clouds. Sc World J 2014;2014.

Specification-Version, O.S., 1.4. 0. 2013, Open Networking Foundation.

Shirali-Shahreza S, Ganjali Y. Efficient implementation of security applications in OpenFlow controller with FleXam. In: IEEE 21st annual symposium on high-performance interconnects (HOTI); 2013.

Tsou T, et al. Use cases for ALTO with software defined networks; 2012.

Vissicchio S, Vanbever L, Bonaventure O. Opportunities and research challenges of hybrid software defined networks. ACM SIGCOMM Comput Commun Rev 2014;44(2):70–5.

Vissicchio S, Vanbever L, Bonaventure O. Opportunities and research challenges of hybrid software defined networks. SIGCOMM Comput Commun Rev 2014;44 (2):70–5.

Wen X, et al. Towards a secure controller platform for openflow applications. In: Proceedings of the ACM second ACM SIGCOMM workshop on Hot topics in software defined networking; 2013.

Wang K, et al. LiveSec: towards effective security management in large-scale production networks. In: Proceedings of the 32nd IEEE international conference on distributed computing systems workshops (ICDCSW); 2012.

Wang J, et al. Towards a security-enhanced firewall application for OpenFlow networks. Cyberspace safety and security. Springer; 92–103.

Wang Y, et al. NetFuse: short-circuiting traffic surges in the cloud. In: Proceedings of 2013 IEEE international conference on communications (ICC); 2013.

Wang C-d, Yu H-f, Wang H-b. Grey self-organizing map based intrusion detection. Optoelectron Lett 2009;5:64–8.

Wasserman M, Hartman S. Security analysis of the open networking foundation (onf) OpenFlow switch specification; 2013.

Woon Hau C, Zhong F, Haines R. Emerging technologies and research challenges for 5G wireless networks. Wirel Commun, IEEE 2014;21(2):106–12.

Xia W, et al. A survey on software-defined networking. Commun Surv Tutor, IEEE 2014;99 1-1.

Xie GG, et al. On static reachability analysis of IP networks. In: Proceedings of INFOCOM 24th annual joint conference of the IEEE computer and communications societies; 2005.

Xing T, et al. Snortflow: a openflow-based intrusion prevention system in cloud environment. In: Second GENI research and educational experiment workshop (GREE); 2013.

Yoo S. Multi-domain cognitive optical software defined networks with market-driven brokers. In: Proceedings of the 40th European conference and exhibition on optical communication. Cannes, France; 2014.

Yang M, et al. Software-defined and virtualized future mobile and wireless networks: a survey. Mobile Netw Appl 2014:1–15.

Yao G, Bi J, Xiao P. Source address validation solution with OpenFlow/NOX architecture. In: Proceedings of the 19th IEEE international conference on network protocols (ICNP); 2011.

Yu M, Jose L, Miao R. Software defined traffic measurement with OpenSketch. In: NSDI; 2013.

You W, et al. Towards security in virtualization of SDN. Int J Comput Control, Quantum Inf Eng 2014;8(8):2014.

Zaalouk A, et al. OrchSec: an orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions. In: IEEE network operations and management symposium (NOMS); 2014.

Zhang Y. An adaptive flow counting method for anomaly detection in SDN. In: Proceedings of the ninth ACM conference on emerging networking experiments and technologies; 2013.