A Survey of Security in Software Defined Networks

Scott-Hayward, S., Natarajan, S., & Sezer, S. (2016). A Survey of Security in Software Defined Networks. IEEE Communications Surveys and Tutorials, 18(1), 623-654. DOI: 10.1109/COMST.2015.2453114

Published in:
IEEE Communications Surveys and Tutorials

Document Version:
Peer reviewed version

A Survey of Security in Software Defined Networks

Sandra Scott-Hayward, *Member, IEEE,* Sriram Natarajan, and Sakir Sezer *Member, IEEE,*

***Abstract*—The proposition of increased innovation in network applications and reduced cost for network operators has won over the networking world to the vision of Software-Defined Networking (SDN). With the excitement of holistic visibility across the network and the ability to program network devices, developers have rushed to present a range of new SDN-compliant hardware, software and services. However, amidst this frenzy of activity, one key element has only recently entered the debate: *Network Security*. In this article, security in SDN is surveyed presenting both the research community and industry advances in this area. The challenges to securing the network from the persistent attacker are discussed and the holistic approach to the security architecture that is required for SDN is described. Future research directions that will be key to providing network security in SDN are identified.**

***Index Terms*—Software Defined Networking, SDN, Network Security, OpenFlow, Secure SDN Architecture.**

I. Introduction

SOFTWARE-DEFINED networking (SDN) has rocketed to the top of the networking agenda since its emergence about 5 years ago. A fundamental characteristic of the SDN architecture is the physical separation of the control plane from the forwarding plane. A logically centralized control function maintains the state of the network and provides instructions to the data plane. The network devices in the data plane then forward data packets according to these control instructions. While this architectural shift has gained significant attention from both the academic and network industry, the concept of separating control and data plane functionality has been around for much longer. In the 1980s, central network control [1] was explored followed by active networks in the 1990s [2] to introduce programmability into the network. During this time, the driving application for the central/programmable network was missing. Then with the arrival of cloud computing and virtualization in the data-center, the right application for SDN was discovered.

One of the proposals for separation of the control and forwarding planes, which led to SDN as it is known today specifically considered the security aspects of such a framework. The SANE architecture [3] centred on a logically centralized controller responsible for authentication of hosts and policy enforcement. At the time of its proposal, this was considered to be an extreme approach that would require a radical change to the networking infrastructure and end-hosts. Ethane [4] then extended the work of SANE but with an approach that required less alteration to the original network. It controlled the network through a centralized controller responsible for enforcing global policy, and Ethane switches that forwarded packets based on rules in a flow table. This simplified network control allowed the data and control plane to be separated to allow for more programmability. Following these early works on security in programmable networks, the focus in the literature transfers to OpenFlow [5]. OpenFlow is an open standard developed as part of Stanford University's clean slate project. The goal of the project was to provide a platform to enable researchers to run experiments in operational networks.

The successful adoption of OpenFlow by both researchers and industry has driven the SDN movement. SDN-enabled networks have already proven to be successful in various deployment scenarios (e.g. Google's backbone network [6], Microsoft's public cloud [7], NTT's edge gateway [8] etc.). In addition, network virtualization software has significantly progressed from trial evaluations to production deployments (e.g. VMWare NSX [9], Nuage Networks VSP [10]). While these trends are promising, one area that has received minimal attention is that of security in SDN.

There are clear security advantages to be gained from the SDN architecture. For example, information generated from traffic analysis or anomaly-detection in the network can be regularly transferred to the central controller. The central controller can take advantage of the complete network view supported by SDN to analyze and correlate this feedback from the network. Based on this, new security policies to prevent an attack can be propagated across the network. It is expected that the increased performance and programmability of SDN along with the network view can speed up the control and containment of network security threats.

On the down-side, the SDN platform can bring with it a host of additional security challenges. These include an increased potential for Denial-of-Service (DoS) attacks due to the centralized controller and flow-table limitation in network devices, the issue of trust between network elements due to the open programmability of the network, and the lack of best practices specific to SDN functions and components. For example, how to secure the communication channel between the network element and the controller when operated in the same trust domain, across multiple domains, or when the controller component is deployed in the cloud?

In the past few years, a number of industry working groups have been launched to discuss the security challenges and solutions. Meanwhile, researchers have presented solutions to some SDN security challenges. These range from controller replication schemes through policy conflict resolution to authentication mechanisms. However, when the extent of the

S. Scott-Hayward and S. Sezer are with the Centre for Secure Information Technologies, Queen's University Belfast, Northern Ireland (e-mail: s.scott-hayward@qub.ac.uk; s.sezer@ecit.qub.ac.uk).

S. Natarajan is with Deutsche Telekom - Silicon Valley Innovation Center (T-Labs), U.S.A. (e-mail: Sriram.Natarajan@telekom.com).

Manuscript received September 30, 2014; revised March 03, 2015 and May 19, 2015; accepted June 26, 2015.

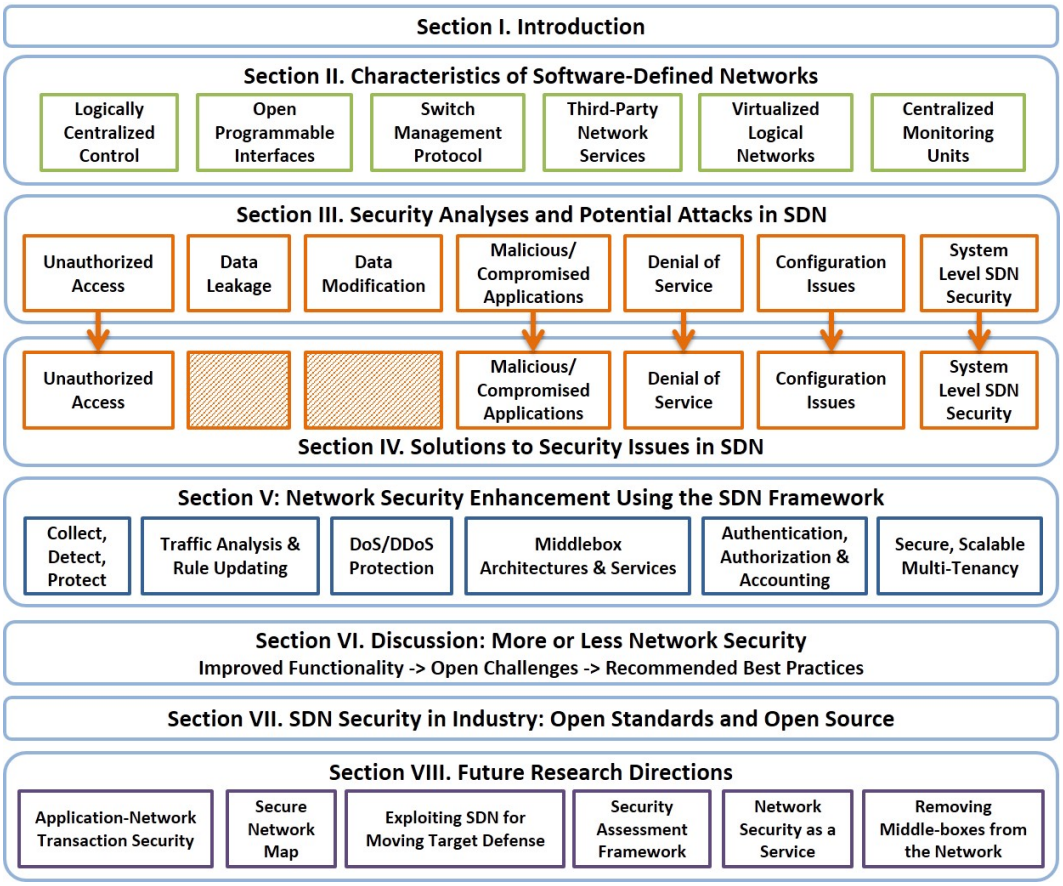


Fig. 1. Overview of the SDN Security Survey

issues is compared to the degree of attention placed on them, it is clear that without a significant increase in focus on security, it is possible that SDN will not succeed beyond the private datacenter or single organization deployments seen today.

The main objective of this paper is to survey the literature related to security in SDN to provide a comprehensive reference of the attacks to which a software-defined network is vulnerable, the means by which network security can be enhanced using SDN and the research and industry approaches to security issues in SDN.

The paper is structured as follows: Section II provides a context to the work by introducing the characteristics of SDN. In Section III recent SDN and OpenFlow security analyses are presented followed by a categorization of the potential attacks to which the architecture is vulnerable. Research work presenting solutions to these attack types is then presented in Section IV. The arrows in Figure 1 indicate the attack categories for which solutions have been proposed and, by extension, those areas which have not yet received research attention. In Section V, the alternative view of SDN security is introduced with a survey of the research work dealing with security enhancements based on the SDN architecture. In Section VI, the two perspectives on SDN security are compared with improved functionality, open challenges, and recommended best practices identified. Section VII highlights open standards and open source industry group work on SDN security. Future research directions are identified in Section VIII. The paper is concluded in Section IX. For clarity, an overview of the Security Survey structure is presented in Figure 1.

II. Characteristics of Software-defined Networks

In this section, the discussion begins with understanding the SDN characteristics in detail. These characteristics are highlighted in Figure 2 and represent the specific features of the SDN framework/architecture that may have an impact on SDN security whether through introducing vulnerabilities or enabling enhanced network security. The 6 characteristics are marked in Figure 2 at the layer/interface/network element that they affect. Potential attacks are introduced in the next Section.

1) Logically Centralized Control: A fundamental characteristic of SDN is the logically centralized, but physically distributed controller component. The controller maintains a global network view of the underlying forwarding infrastructure and programs the forwarding entries based on the policies defined by network services running on top of it. While early controller developments (e.g. NOX [11], Beacon [12], Floodlight [13]) were structured around functioning as an OpenFlow [5] driver, various new implementations (e.g. OpenDaylight [14], OpenContrail [15]) have matured to provide the required abstractions to the network services and to support mul-

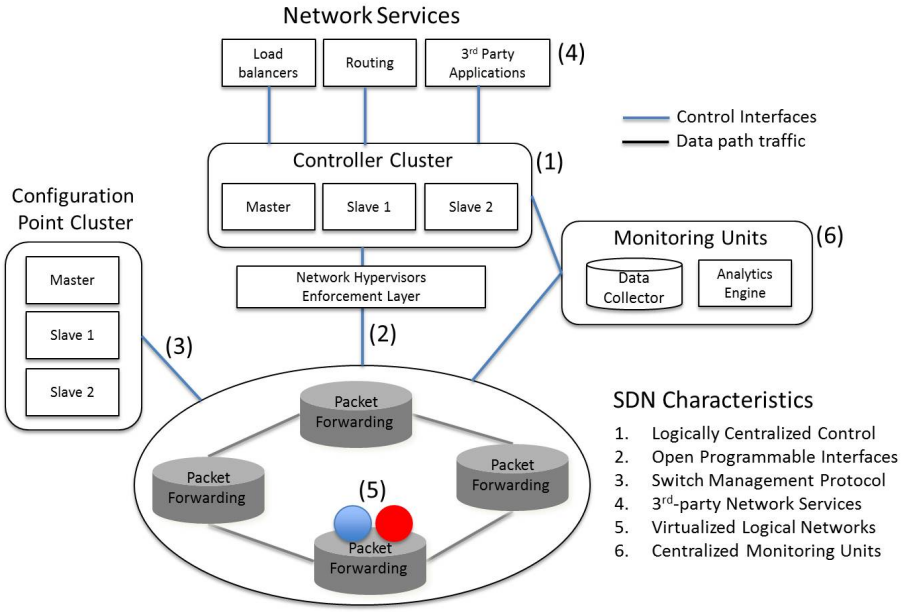


Fig. 2. SDN Characteristics

tiple programming interfaces (e.g. NETCONF, XMPP, BGP) to manage the forwarding devices.

Similarly, evolving from a single controller design, several options for distributed control (controller cluster) have been proposed for scalability and reliability requirements, as shown in Figure 3. Distributed control with multiple controller instances is proposed in Onix [16], SoftCell [17], HyperFlow [18], and Kandoo [19]. These approaches will be discussed in Section IV. ONOS [20] and OpenDaylight [14] implement distributed control with multiple instances forming a cluster as illustrated in Figure 3. In each case, each individual controller instance is the exclusive master of a set of switches and the controllers are clustered in Master/Slave groups.

2) Open Programmable Interfaces: Unlike traditional networking equipment, SDN physically separates the control and data plane entities. The primary motivation with this characteristic is to simplify the forwarding devices and allow the networking software in the controller to evolve independently. This functionality introduces the potential for innovation and easier adoption of novel solutions. A standardized programmable interface, OpenFlow [5], was adopted by the industry in order to program multiple flavors of forwarding devices (i.e. ASIC, FPGA-based, Network Processors, virtual switches) thereby abstracting the complexity of the underlying hardware. Several interfaces are identified in Figure 4: the Control-Data Interface (also known as the Southbound API such as OpenFlow, OF-Config, OVSDB, NETCONF), the Application-Control Interface (also known as the Northbound API such as REST API) and the East-West Interface between Controllers. The East-West interface refers to the bidirectional and lateral communication between SDN controllers. These controllers may belong to the same or different SDN control domains. The east/westbound APIs for this interface are discussed in [21]. It is noted that in [22], Jarschel et al. propose a definition referring to the east interface for communication between SDN controllers and the west interface for communication between an SDN controller and other, non-SDN control planes. The interoperability between SDN and legacy control planes is, however, out of scope of this work.

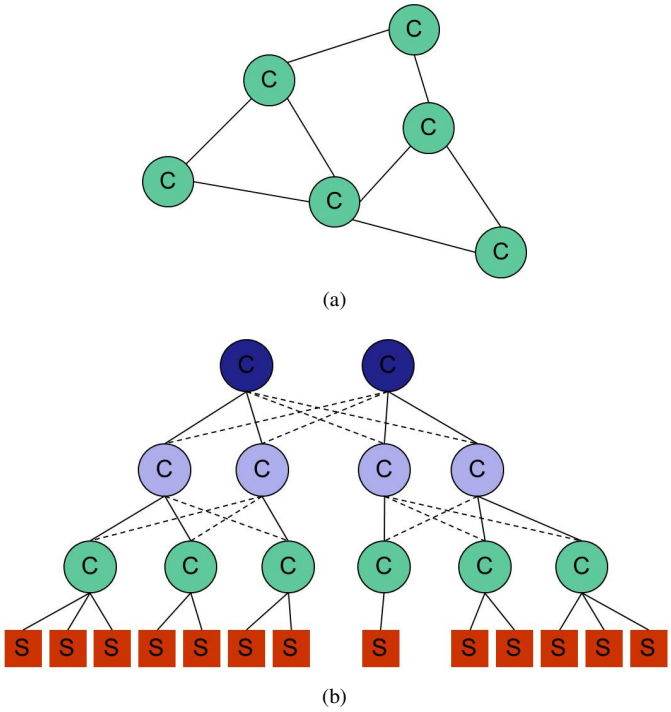


Fig. 3. Distributed Control Frameworks for SDN (a) Controller Clustering, and (b) Hierarchical Control

3) Switch Management Protocol: A companion interface to the programmable interface described above is the switch management protocol (e.g. OF-Config, OVSDB [23]). Such a protocol is required to standardize the configuration and management functions of the programmable hardware. For instance, the OF-Config protocol is used to configure and manage an OpenFlow capable switch as well as multiple logical switches that can be instantiated on top of the device. Internally, the protocol uses NETCONF as the transport protocol that defines the set of operations over a messaging layer (RPC), which exchanges the switch configuration information between the configuration point and the packet forwarding entity (i.e. (3) in Figure 2).
4) Third-party Network Services: SDN allows the integration of third-party network services in the architecture. In a monolithic SDN controller implementation (e.g. RYU, POX, NOX), these applications are compiled and run as part of the controller module while controllers like OpenDayLight allow the instantiation of applications at run-time, without restarting the controller module. This is analogous to operating systems, wherein software modules and libraries can be downloaded and integrated within a running environment. From a deployment standpoint, this drives innovation, allows customization of services, introduces flexibility in the overall architecture to adapt to new features, and reduces the cost of proprietary services. Depending on the controller implementation, third-party services can communicate to a controller module via internal APIs or open northbound APIs (e.g. REST APIs) supported by the controller.
5) Virtualized Logical Networks: Virtualizing the SDN components supports multi-tenancy in the infrastructure. In a typical SDN network, multiple logical switches can be instantiated in a shared physical substrate such that each entity can represent individual tenants/customers. The goal here is to containerize the SDN components thereby guaranteeing customized performance, security, and Quality of Service (QoS) based on tenant requirements. While SDN is developing in the IT community, Network Functions Virtualization (NFV) is being developed by the Telecommunications industry. NFV uses IT virtualization technologies to virtualize network functions/services previously implemented in proprietary hardware appliances. This supports dynamic and agile network service provision. NFV and SDN are closely connected offering a software-based networking paradigm.
6) Centralized Monitoring Units: Although not unique to the SDN architecture, a centralized monitoring unit unifies the analytical capabilities of the infrastructure and creates a feedback control loop with the controller to automate updates to the networking function. For example, a TAP monitoring unit can feed data traffic to Deep Packet Inspection (DPI) engines that can assess the data traffic, identify attack patterns and then programmatically update the forwarding table to block attack traffic. (*Note:* For illustration purposes, the monitoring units are separated from the controller in Figure 2. It is equally possible that this functionality be collocated with the controller.) While the SDN entities can internally include several monitoring capabilities, a typical network deployment would consider deploying dedicated monitoring solutions in the infrastructure. For example, the OpenFlow protocol provides statistical and status information about the switch and its internal state (e.g. the flow state maintained in the Flow Table, the status of the ports and links, statistical information on flows, ports, queues, and meters). These are inherent monitoring functions that are part of the underlying architecture components. As discussed before, a practical deployment can also deploy visibility tools and solutions like sFlow, NetFlow, or integrate third party visibility fabric for monitoring purposes. The monitoring logic that is part of the feedback loop is responsible for comprehending the collected information and updating the controller for required updates to be provided to the network devices.

If the features in Figure 2 are simplified to a set of layers and interfaces as shown in Figure 4, the challenges associated with each layer of the framework and on the interfaces between the layers can be identified. This framework is used throughout the survey to categorize both the challenges and the proposed solutions for SDN security.

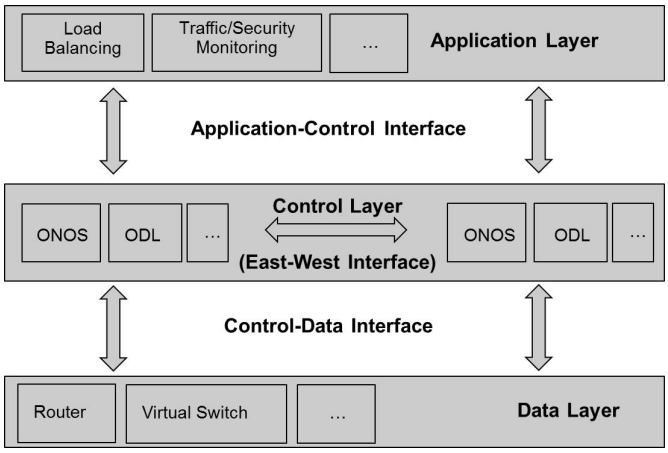


Fig. 4. SDN Functional Architecture illustrating the data, control and application layers and interfaces

III. Security Analyses and Potential Attacks in SDN

Given the SDN characteristics detailed in Section II, existing security analyses in SDN are discussed next and the potential security issues in the system are then classified.

A. Analyses of SDN Security

A number of SDN/OpenFlow security analyses precede this survey [21], [24]–[34]. The first of these references is our initial contribution on this subject of SDN security [24]. This current document significantly expands on the original

contribution with detailed discussion of the SDN characteristics and potential attacks, security issues and enhancements, analysis of the industry work on SDN security, and inclusion of recommended best practices and future research directions.

In the case of each analysis work [21], [24]–[34], the authors found that the altered elements or relationship between elements in the SDN framework introduce new vulnerabilities, which were not present before SDN. However, the more recent analyses [24], [30]–[34] also identify security enhancements introduced by SDN.

OpenFlow is currently the most commonly deployed SDN technology. As a result, a number of the security analyses focus only on OpenFlow. In [25] an analysis of the OpenFlow protocol using the STRIDE threat analysis methodology is completed [35]. The work focuses on the execution of Information Disclosure and DoS attacks, which the author established could be successfully executed. Although a number of mitigation techniques are proposed, such as aggregating flow rules to reduce the flow table size, these techniques are not proven in the work.

The OpenFlow switch specification [36] describes the use of Transport Layer Security (TLS) with mutual authentication between the controllers and their switches. However, the security feature is optional, and the standard of TLS is not specified. The lack of TLS adoption by major vendors (and in the majority of open-source controllers and switches) and the possibility of DoS attacks leading to fraudulent rule insertion and rule modification is discussed in [26].

In [27] a high-level analysis of the overall security of SDN is presented with the conclusion that new responses are required to the new threats arising from centralized control and network programmability. Three of the seven identified threat vectors are specific to SDN and relate to controller software, the control-data interface, and the control-application interface. A number of solution techniques are proposed. For example, replication of controllers and applications to provide alternative management/control in the case of hardware or software faults, diversity of controllers for robustness against software bugs and vulnerabilities in any single controller, and secure components such as trusted computing bases (TCB) to protect the confidentiality of sensitive security data. A comprehensive survey of SDN [21] has also been co-produced by the authors of [27]. The security and dependability of SDN are reviewed with a specific focus on the security of OpenFlow and countermeasures for security threats in OpenFlow networks.

The vulnerability of the SDN network to attack has also been tested [28], [29]. In [28], the research network and testbed, ProtoGENI, has been analyzed. It was discovered that numerous attacks between users of the testbed along with malicious propagation and flooding attacks to the wider internet were possible when using the ProtoGENI network. More recently, a preliminary work considered the feasibility of a fingerprinting attack against an SDN network [29]. In the attack, the network is first fingerprinted to determine whether it uses SDN/OpenFlow switches. The SDN network is then subjected to a DoS attack on the control plane via the data-control plane communication path and on the data plane via the flow table of the network device. A DoS attack refers to an attempt to make a machine or network resource unavailable to its intended users. In the case of SDN, the network devices require access to the control plane to receive traffic management instructions and traffic across the network requires access to the network device flow tables to dictate traffic management policies. The data-control plane interface and the network device flow table are therefore points of vulnerability to DoS attack.

The extent to which network security management is improved in SDN-based networks is discussed in [30]. The discussion is split by infrastructure, software stack, and network protocols with the majority of the discussion focussed on protocol security, which the authors define as confidentiality and authentication. They conclude that security of the switch-controller and controller-controller communication protocols require further work. The security of switch-controller communication has inspired several research works presented later in this article and, indeed, the industry standardization bodies security group work. Consideration of controller-controller communication security is less well explored and an important consideration for wide deployment of SDN.

In [31], an evaluation methodology is presented for the security *of* SDN and security *by* SDN as compared to conventional networks. To evaluate the security of SDN (i.e. security of the reference architecture), the criteria of confidentiality, authenticity, integrity, availability, and consistency are used. To evaluate the security by SDN (i.e. benefits provided by SDN), the criteria are: network management, costs, and attack detection and mitigation. The conclusion is that the benefits outweigh the drawbacks. Although the authors argue that many of the threats identified in SDN also exist in conventional networks, they do not explore the greater attack surface exposed by the layered architecture of SDN and the impact of this on the overall vulnerability of the architecture. As in [30], security challenges related to inter-provider SDN deployments are identified for future consideration.

In a combination of a security analysis and a framework proposal in [33], the authors recommend the use of SDN for security enhancement in wireless mobile networks. The work includes a limited survey of SDN-based security solutions categorized by target environment. Several requirements are identified for the SDN security design; interoperability, responsiveness, compatibility, adaptation, and simplicity. A framework with a wireless mobile security layer above the control plane interacting with local agents at the data plane is proposed to meet these requirements.

The close link between SDN and NFV has been highlighted with respect to the *Virtualized Logical Network* characteristic of SDN described in Section II. In [34], the impact of SDN on cloud security and the potential risks introduced when SDN is deployed within and across clouds are discussed. The opportunities and vulnerabilities are mixed in the discussion. However, a clear observation is made; that the expansion of SDN deployments from campus networks to wide area networks will require increasingly complex security considerations to be incorporated into SDN design. This stems from securely exposing SDN programmability and providing

TABLE I
COMPARISON OF SECURITY ANALYSES OF SDN AND OPENFLOW

Research Work	Security Analysis		OF	SDN Layer/Interface				
	Vulnerabilities	Enhancements		App	App-Ctl	Ctl	Ctl-Data	Data
SDN Security Survey [24]	✓	✓		✓	✓	✓	✓	✓
OF Security [25], OF Vulnerability [26], ProtoGENI [28]	✓		✓			✓	✓	✓
Secure and Dependable SDN [27]	✓			✓		✓	✓	✓
Comprehensive Survey [21]	✓		✓	✓		✓	✓	✓
Attacking SDN [29]	✓		✓			✓	✓	✓
Vulnerability of FlowVisor [32]	✓		✓			✓	✓	
SDN for Network Security [30]	✓	✓	✓		✓	✓	✓	
Blessing or Curse? [31]	✓	✓	✓			✓	✓	✓
SDN Wireless Mobile [33]	✓	✓	✓	✓		✓	✓	✓
Cloud Computing Security [34]	✓	✓				✓	✓	✓

appropriate trust boundaries to secure each layer of the SDN model. In Section VIII of this work, future research directions will be discussed identifying approaches to optimize security and performance of SDN.

The security analyses described in this section are comparatively presented in Table I in terms of their consideration of OpenFlow and the SDN Layer/Interfaces discussed (Figure 4). As previously mentioned, the focus of many of these analyses is the OpenFlow protocol and a framework that supports applications to modify the network state in the forwarding device via OpenFlow protocol. As a result, the majority of issues presented in this Section relate to control and data plane security and the control-data interface. A few of these analyses [21], [24], [27] specifically highlight the security issue of introducing 3rd party applications to the network. In this paper, a deeper analysis of these issues is provided (Section IV).

B. Attacks and Vulnerabilities in SDN

To elaborate the security analysis work, the SDN security issues are categorized by type and with respect to the SDN layer/interface affected by each issue/attack. As shown in Table II, the issues are split into seven main categories and provide a number of specific examples of the way in which the security issue might arise. Only network security issues or attacks specific to the SDN framework are detailed. Details of the attack scenarios are provided in the following subsections and the impacted entities in the architecture are highlighted in Table II. The security issues identified in Table II are also mapped to the SDN architecture in Figure 5 to highlight the entity and interface impacted by the attack or vulnerability. It can be noted that several of these attacks relate directly to the characteristics identified in Section II e.g. Potential Attack - *Unauthorized Access* links to Characteristic - *Logically Centralized Control*. These links will be identified in the subsequent sections.

1) Unauthorized Access: This category relates to access control. One of the original characteristics of SDN is described as centralized control. With the evolution of the technology, this is more accurately described as logically centralized control, which in many implementations is distributed. In the functional architecture of SDN, it is therefore possible for multiple controllers to access the data plane of the network. Similarly, applications from multiple sources (3rd party apps) may link to a pool of controllers. The controller provides an abstraction to applications so that the applications can read/write network state, which is effectively a level of network control. If an attacker impersonated a controller/application, it could gain access to network resources and manipulate the network operation.

2) Data Leakage: There are a variety of potential actions described in the OpenFlow switch specification [36] for packet handling. These include forward, drop and send to controller. It is possible for an attacker to determine the action applied to specific packet types by means of packet processing timing analysis. For example, the time to process a packet passed directly from input port to output port will be shorter than for a packet to be redirected to the controller for processing. The attacker can therefore discover the proactive/reactive configuration of the switch. With a further set of crafted packets, an attacker could infer additional information about the network device. Having discovered the packet type that is redirected to the controller, the attacker can then generate a volume of fake flow requests leading to a Denial of Service (DoS) attack. The relationship between this type of data leakage to the DoS attack is illustrated in [29].

Another open challenge in the SDN architecture is the secure storage of credentials (e.g. keys and certificates) for multiple logical networks in the programmable data plane. Previously, cross-VM (virtual machine) side channel attacks have been demonstrated in the cloud environment. In that attack, a malicious VM identifies a vulnerable VM and extracts secure information from the targeted resource [37]. Similar data leakages are possible in the SDN environment. For example, OF-Config instantiates multiple OF-Logical switches on-top of an OpenFlow capable switch. Assume each logical entity is assigned to a different customer. Then, if the logical networks and their associated credentials (e.g. keys, certificates) are

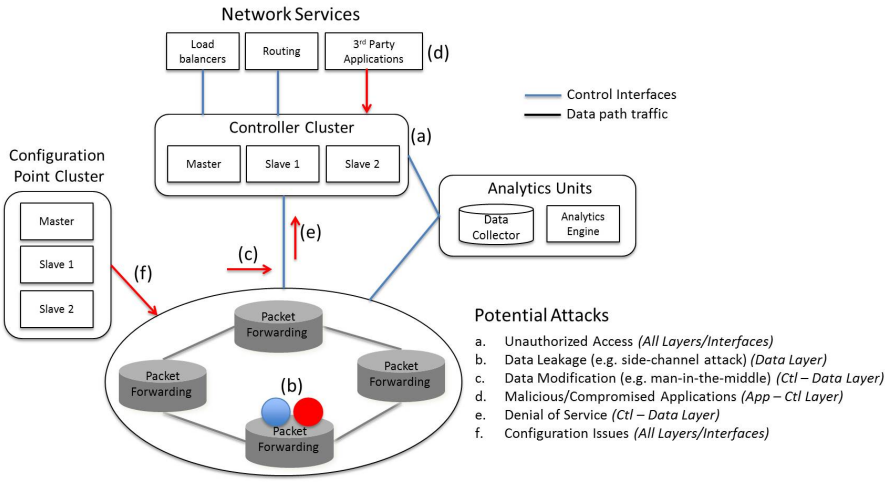


Fig. 5. SDN Potential Attacks and Vulnerabilities

not securely isolated or containerized, this can lead to data leakages that can compromise the functionality of the logical network instance.

3) Data Modification: As previously mentioned, the controller has the ability to program the network devices to control the flow of traffic in the SDN. If an attacker is able to hijack the controller then it would effectively have control over the entire system. From this privileged position, the attacker can insert or modify flow rules in the network devices, which would allow packets to be steered through the network to the attacker's advantage.

Similarly, if intermediate entities are introduced between the data and control plane components for provisioning of virtual networks (e.g. FlowVisor [38], OpenVirtex [39], [40], FlowN [41], Layer 2/3 agents etc.), then proper security mechanisms should be used between every interface, component, and communication channel [8]. With regards to the communication channel, the OpenFlow switch specification [36] describes the use of TLS with mutual authentication between the controllers and their switches. However, the security feature is optional, and the version of TLS is not specified. The lack of TLS adoption by major vendors (and in the majority of open-source controllers and switches) can allow a man-in-the-middle attacker to impersonate the controller and launch various attacks; for example, manipulate the control (e.g. OpenFlow) messages sent by the controller or inject reset messages to tear down the connection. In addition, the intermediate components between the control and data plane should be secure enough to avoid the introduction of additional security issues. A man-in-the-middle attack occurs when the attacker has the ability to intercept messages between two victims to monitor and alter or inject messges into the communication channel. This is possible when there is no authentication of the communication endpoints. FlowVisor [38], a network hypervisor for OpenFlow was identified to lack a proper isolation mechanism [32], which can allow an attacker to launch data modification attacks on the communicating entities. The issue of data modification is a specific concern in the split-plane architecture of SDN.

4) Malicious/Compromised Applications: Given that the controller acts as an abstraction from the data plane for the applications and that SDN enables 3rd party applications to be integrated into the architecture [42], a malicious application could have as much of a detrimental effect on the network as a compromised controller. Similarly, a poorly designed or buggy application could unintentionally introduce vulnerabilities to the system. For example, a known bug could be exploited by an attacker to drive the application into an unsafe state. This issue has been considered by a few researchers and will be discussed in Section IV.

5) Denial of Service: One of the core security weaknesses of SDN is introduced by the SDN architecture; the combination of the central controller and separation of the control and data planes. Due to the communication path between the controller and the network device, an attacker could flood the controller with packets requiring a flow rule decision and render it unavailable for legitimate users. A similar DoS attack could be performed at the infrastructure level with flooding of the flow table for which limited memory resources are available [43]. In addition, the possibility of DoS attacks leading to fraudulent rule insertion and rule modification is discussed in [26]. This attack vector has received considerable attention from the research community, as will be seen in Sections IV and V.

6) Configuration Issues: Network security policies and protocols are continuously developed as network vulnerabilities are detected. Many of these will apply to the layers and interfaces of the SDN framework. However, there is little protection from such policies if they are not implemented or they are disabled without understanding the security implications of the deployment scenario. In an SDN-based network, it will be important for network operators to enforce the implementation of policies such as TLS. As highlighted in Table II, misconfigurations or incorrect use of security features can impact all layers in the architecture. As discussed in Section III-B3, disabling a secure connection can adversely impact the network functions by introducing several potential attacks in the framework.

TABLE II
CATEGORIZATION OF THE SECURITY ISSUES ASSOCIATED WITH THE SDN FRAMEWORK BY LAYER/INTERFACE AFFECTED

Security Issue	SDN Layer Affected or Targeted				
	App Layer	App-Ctl Interface	Ctl Layer	Ctl-Data Interface	Data Layer
Unauthorized Access e.g.					
Unauthorized Controller Access/Controller Hijacking			✓	✓	✓
Unauthorized/Unauthenticated Application	✓	✓	✓		
Data Leakage e.g.					
Flow Rule Discovery (Side Channel Attack on Input Buffer)					✓
Credential Management (Keys, Certificates for each Logical Network)					✓
Forwarding Policy Discovery (Packet Processing Timing Analysis)			✓	✓	✓
Data Modification e.g.					
Flow Rule Modification to Modify Packets (Man-in-the-Middle attack)			✓	✓	✓
Malicious/Compromised Applications e.g.					
Fraudulent Rule Insertion	✓	✓	✓		
Denial of Service e.g.					
Controller-Switch Communication Flood			✓	✓	✓
Switch Flow Table Flooding					✓
Configuration Issues e.g.					
Lack of TLS (or other Authentication Technique) Adoption	✓	✓	✓	✓	✓
Policy Enforcement	✓	✓	✓		
Lack of Secure Provisioning	✓	✓	✓	✓	✓
System Level SDN Security e.g.					
Lack of Visibility of Network State			✓	✓	✓

Opening the interfaces between network components has the potential to introduce considerable vulnerabilities; not only regarding interoperability between multiple vendor devices, but also for data/control communication across these new interfaces. SDNs provide us with the ability to easily program the network and to allow for the creation of dynamic flow policies. It is, in fact, this advantage that may also lead to security vulnerabilities. With policies from multiple applications or installed across multiple devices, inconsistencies must be detected to resolve policy conflicts. Several solutions have been proposed in the literature and will be introduced in Section IV.

7) System Level SDN Security: At a system level, there are also a number of security concerns in SDN. A major industry concern is satisfaction of the auditing process. It is vital, in terms of network compliance and operation, to be able to provide a controlled inventory of network devices. This involves knowledge of what devices are running, how they are bound to the network etc. For example, OpenFlow switches can operate in either a *fail-secure mode* or *fail standalone mode*. When the switch is disconnected from the controller, the switch's internal logic can choose to operate in one of these modes. Upon re-connection, the controller takes control of the switch and its internal network state. From an operational perspective, it is critical for an operator to understand the mode in which the switch operated during connection interruption, the forwarding behavior during failures, the impacted flow entries, and the behavior of the controller after reestablishing the connection. For accountability and auditing, the ability to retrospectively retrieve such operational information is critical and should be managed in SDN.

Network security is a challenging task. In order to provide correct access to resources, security professionals are tasked with implementing a large range of techniques that increase the complexity of the network and make it difficult to manage. The basic properties of a secure communications network are: confidentiality, integrity and availability of information. These properties are supported by techniques of authorization, authentication and encryption. The sum of these properties presents a network in which the data, the network assets (e.g. devices) and communication transactions are secure and protected from either malicious attack or unintentional damage. This section exhibits some of the open challenges and possible vulnerabilities in a SDN-enabled network. Therefore, it is critical to entail protection in the architecture for a secure functioning of the network infrastructure.

IV. SOLUTIONS TO SECURITY ISSUES IN SDN

Seven categories of security issue/attack type were identified in Table II. In the process of categorizing the solutions presented in the literature (as shown in Table III), it is found

TABLE III
COMPARISON OF RESEARCH ON SOLUTIONS TO SECURITY ISSUES IN SDN

Solution to Security Issue	Research Work	SDN Layer/Interface				
		App	App-Ctl	Ctl	Ctl-Data	Data
Unauthorized Access	Securing Distributed Control [44], Byzantine-Resilient SDN [45]			✓	✓	
	Authentication for Resilience [46]			✓		
	PermOF [47]	✓	✓			
	OperationCheckpoint [48]	✓	✓	✓		
	SE-Floodlight [49], [50]	✓	✓	✓	✓	
	AuthFlow [51]	✓		✓	✓	✓
Data Leakage						
Data Modification						
Malicious Applications	FortNOX [52]	✓	✓	✓	✓	
	ROSEMARY [53]	✓		✓		
	LegoSDN [54]	✓	✓	✓		
Denial of Service	AVANT-GUARD [55], CPRecovery [56]			✓	✓	✓
	VAVE [57]	✓		✓	✓	✓
	Delegating Network Security [58]	✓	✓	✓	✓	✓
Configuration Issues	NICE [59]	✓	✓		✓	
	FlowChecker [60], Flover [61], Anteater [62], VeriFlow [63], NetPlumber [64]	✓	✓	✓	✓	
	Security-Enhanced Firewall [65], FlowGuard [66], [67], LPM [68]	✓		✓	✓	✓
	Frenetic [69], Flow-Based Policy [70], Consistent Updates [71]	✓	✓	✓	✓	
	Shared Data Store [72]	✓		✓	✓	✓
	Splendid Isolation [73]		✓	✓		
	Verificare [74], Machine-Verified SDN [75], VeriCon [76]		✓	✓	✓	
System Level SDN Security	Debugger for SDN [77]	✓			✓	
	OFHIP [78], Secure-SDMN [79]				✓	
	FRESCO [80]	✓	✓	✓	✓	

that solutions are proposed in only 5 of the 7 categories. There are no proposed solutions to date for *Data Leakage* or *Data Modification* issues.

Table III presents both a categorization of the research work by security issue (corresponding to Table II), and a comparison of each research work identified within these categories against the relevant SDN layer/interface. An immediate observation from the table indicates that the categories of *Unauthorized Access* and *Configuration Issues* have received most attention to date. It can also be identified that the solutions impact all layers/interfaces of the SDN infrastructure. However, the data layer is least affected. This is due to the strong focus on software solutions. Further detail on each solution is provided in the subsequent sections.

A. Unauthorized Access

Two attack vectors relating to unauthorized access were raised in Table II; unauthorized controllers and unauthorized applications. The issue relates to the potential damage that could be wreaked on the network by either unauthorized SDN element reconfiguring the network.

In an attempt to remove the controller bottleneck and improve network efficiency, the authors of [44] describe a hybrid control model. Control is centralized under normal circumstances but in the case of heavy load, network equipment assumes the job of flow rule installation on other network equipment on behalf of the controller. In [44], the method to secure the distributed control model is presented. It is assumed that the central control element is secured by TLS. The authors then present a signature algorithm to securely transmit flow installation requests from network device to network device. The system requires a centralized trust manager and introduces significant overhead in message-passing and signature-checking.

The vulnerability of the SDN controller to attack is identified in [45]. The authors present a secure SDN structure with each network element managed by multiple controllers using the Byzantine mechanism. The number of controllers used to support the Byzantine fault tolerance is minimized with a cost-efficient controller assignment algorithm.

Several distributed control designs have been proposed for scalability and reliability, as identified in Section II [17]–[19], [81]. Their features are highlighted here as the distributed mechanisms inherently increase the protection of the network against the effect of unauthorized controller access. In HyperFlow [18], a publish/subscribe mechanism is used to synchronize the network state between a group of distributed controllers. A HyperFlow controller publishes events that alter the system state and the other controllers replay all published events to reconstruct the state. The control plane response time

is reduced by the localized decision-making. With SoftCell [17], the authors use local software agents to cache packet classifiers and policy tags so that the main controller load is reduced. In Kandoo [19], local decision-making is separated from network-wide decision-making. Certain applications can be supported by event processing at local controllers reducing the load on the root controller. A proposal for using building blocks - the *Logical xBar* and the Logical Server in recursive aggregation is presented in [81]. This enables local control at the appropriate level of the hierarchy and is proposed to extend SDN to large-scale networks. Although these designs are not specifically security-focussed, the resilience introduced by the ability to delegate and/or distribute control can mitigate the vulnerability of the individual SDN controller to attack. While several open-source controllers (e.g. ONOS [20] and OpenDaylight [14]) implement distributed control, there are additional measures required for a fully secure, robust, and resilient SDN control plane, as detailed in [82].

From a deployment perspective, an additional concern to unauthorized access stems from the lack of TLS adoption in real-world deployments [26]. This should be an important consideration when the applications, controllers and network elements are deployed across trust domains (e.g. the Internet).

Both authentication and authorization of the applications is required to ensure that only trusted applications should connect to the network. This issue is highlighted in [46]. In order to reduce the number of points of serious failure, an SDN setup with a hierarchical system of controllers and switches is proposed. By dividing work across layers of "middle management" controllers between a root controller and the switch fabric, the potential damage of unauthorized access can be localized. This approach assumes that the root controller is trusted. With this architecture, it might be possible to limit the impact of a malicious/compromised application as the application code would run at a "middle management" controller with appropriate protection. However, while the authors emphasize the need for urgent research to tackle the protection issues of SDN deployment in large heterogeneous networks, they do not propose how this protection should be achieved.

A fault-tolerant controller architecture is presented in [83]. In order to guarantee a smooth transition between controller instances in the case of a failure, the data store from which the controllers derive the network state is implemented as a fault-tolerant replicated state machine. Communication between the controller instances must be secured.

The issue of exposing the full privilege of OpenFlow to every application without protection is identified in [47]. The authors propose *PermOF* with a set of permissions and an isolation mechanism to enforce the permissions at the Application Programming Interface (API) entry. The solution effectively applies minimum privilege on the applications protecting the network from control-plane attacks.

The concept of the permissions system is extended in [48]. *OperationCheckpoint* is designed and implemented on the Floodlight controller [13]. The authors define the set of permissions to which the application must subscribe on initialization with the controller and introduce an *OperationCheckpoint*, which implements a permissions check prior to authorizing application commands. An unauthorized operations log is used to audit malicious activity to build a profile for SDN application-layer attacks.

SE (Security Enhanced) Floodlight [49], [50] developed by Stanford Research Institute is an extension to the Floodlight OpenFlow controller. SE-Floodlight introduces a security enforcement kernel (SEK), which is an improvement of FortNOX (see Section IV-B) and includes a digitally authenticated northbound API. An administrator is required to pre-sign the OpenFlow application's java class, which may be digitally verified by the SEK at runtime. Once signed and validated, the application has permission to modify or query the network, or traffic on the network. The distinction between SE-Floodlight and PermOF and OperationCheckpoint is in the granularity of the approach. PermOF and OperationCheckpoint allow a set of actions to be granted to an application rather than validating the complete application.

To deny access to the SDN by unauthorized hosts, AuthFlow, an authentication and access control mechanism based on host credentials, is proposed in [51]. AuthFlow is implemented with an OF controller, an authenticator, and a RADIUS server. The authenticator intercepts Extensible Authentication Protocol (EAP) messages between the requesting host and the RADIUS authentication server and provides an authentication response to the OF controller. The controller allows or denies traffic based on the authentication response. Access control is implemented by pairing host credentials with a set of host flows. The solution presented by AuthFlow challenges the SDN security issue of unauthorized access.

Unauthorized access to critical SDN elements is being tackled at all layers of the SDN architecture. With a combination of the described solutions, a defense-in-depth approach to securing the SDN with layers of trust could be achieved.

In Table IV, a summary of the problem/goal and the solution proposed for each research work presented under *Unauthorized Access* Issues in SDN is provided. In [44]–[46], solutions to protect against unauthorized controller access are proposed. With [47]–[49], the solutions are designed to protect against unauthorized applications, and in [51], a host-level access control solution is proposed.

B. Malicious/Compromised Applications

At a minimum, in order to avoid the deployment of malicious/compromised applications, controllers and applications should establish a trusted connection and authenticate the identity of the entities before exchanging control messages. Several more advanced solutions are presented in [52]–[54].

A security enforcement kernel is proposed in [52] as a solution to the problem of malicious controller applications. FortNOX implements role-based authentication for determining the security authorization of each OF application. The FortNox enforcement engine handles possible conflicts with flow rule insertion whereby rule acceptance/rejection is dependent on the author's security authorization. A new flow rule that conflicts with an existing flow rule will be detected by FortNOX. If the new (conflicting) flow rule request was generated

TABLE IV
PROBLEM AND SOLUTION PROPOSED FOR *Unauthorized Access* ISSUES IN SDN

Research Work	Problem/Goal	Proposed Solution
Securing Distributed Control [44]	Secure the distributed control model against malicious use	Signature algorithm to securely transmit flow installation rules
Byzantine-Resilient Secure SDN [45]	Protect the SDN Control Plane from attack	Multiple controller structure with Byzantine-Fault Tolerant algorithm
Authentication for Resilience [46]	How to structure the SDN architecture to offer more security?	Hierarchical System of controllers/switches to reduce points of serious failure
PermOF [47]	Full privilege of OF exposed to every application	Proposed Permission System to apply minimum privilege to applications
OperationCheckpoint [48]	Controller operations open to every application	Implementation of a Permissions Check Mechanism to secure app-control interface
SE-Floodlight [49], [50]	Lack of security between OF apps/modules and control/data plane communication	Role-based authorization and security constraint enforcement for OF control layer
AuthFlow [51]	Prevent access to the SDN by unauthorized hosts	An authentication and access control mechanism based on host credentials

TABLE V
PROBLEM AND SOLUTION PROPOSED FOR *Malicious/Compromised Application* ISSUES IN SDN

Research Work	Problem/Goal	Proposed Solution
FortNOX [52]	Challenge of detecting and reconciling potentially conflicting flow rules from OF apps	Security enforcement kernel for prioritizing flow rules with role-based authorization
ROSEMARY [53]	Protect against simple/common network app failures leading to loss of network control	A controller implementing a network app containment and resilience strategy
LegoSDN [54]	Make the controller and network resilient to SDN application failures	A controller architecture to improve availability with fault isolation and network transaction management

by a higher priority author, then the existing flow rule will be replaced. However, if the new flow rule is produced by a lower priority author, then it will be ignored. Three flow rule producer roles are defined; OF Operator, OF Security, and OF Application. A limitation of this approach is the determination of appropriate security authorization level. FortNOX does not resolve the issue of application identification and priority enforcement.

With ROSEMARY [53], the authors propose a robust, secure and high-performance network operating system (NOS). The central objective of ROSEMARY is to improve the resilience of the control plane to both buggy and malicious applications. To achieve this, the authors propose a micro-NOS architecture. Each OF application is run within an independent instance of ROSEMARY effectively sandboxing the application to protect the control layer from any vulnerability or malicious operation of the application. The solution separates network applications from the trusted computing base of the NOS, monitors and controls network resources consumed by each application, monitors and controls application operations such as privileged system calls, and implements a safe NOS restart process to carefully restart each service. This is the most advanced work to date in the protection against malicious/compromised SDN applications.

With LegoSDN [54], the authors consider the consequence of SDN application failures on the controller reliability. In order to avoid the crash of an SDN application leading to a crash of the SDN controller, the authors propose an isolation layer between SDN-Apps, a network-wide transaction system to support atomic updates and efficient rollbacks, and a fault-tolerance layer to detect and overcome crash triggering events. As with ROSEMARY, the dual concerns of security and availability motivate LegoSDN.

One of the SDN characteristics identified in Section II is *Third-party Network Services*. The potential to integrate third-party applications into the SDN via the SDN controller increases the concern for controller/NOS stability in the face of buggy and malicious applications. ROSEMARY [53] and LegoSDN [54] are valuable proposals to protect against such threats. However, significant further research is required to optimize these solutions for a broad spectrum of failure types while maintaining performance requirements of wide-scale SDN deployments.

In Table V, a summary of the problem/goal and the solution proposed for each research work presented under *Malicious/Compromised Application* Issues in SDN is provided. [52] enforces application authorization to protect the network from malicious/compromised applications while [53]

TABLE VI
PROBLEM AND SOLUTION PROPOSED FOR *Denial of Service* ISSUES IN SDN

Research Work	Problem/Goal	Proposed Solution
AVANT-GUARD [55]	Protect against Control Plane DoS attack and detect and respond to changing flow dynamics	Connection Migration Tool reducing data-control plane interaction and Actuating Trigger to install flow rules
CPRecovery [56]	Protect centralized network OS from failure due to DoS attack	CPRecovery component provides seamless primary controller backup
VAVE [57]	Source Address Validation	NOX controller determines validation rules with global view
Delegating Network Security [58]	Remove network administration bottleneck	ident++ protocol to delegate aspects of network security policy

and [54] isolate the applications to protect the controller and the network.

C. Denial of Service

As described in Section III, there is a specific weakness introduced in SDN by the separation of the control and data plane, which can lead to a DoS attack on the controller or on the switch flow table. A number of solutions have been proposed to overcome this weakness [55]–[58]

AVANT-GUARD [55] proposes a solution to the control-data plane communication bottleneck in SDN by limiting the flow requests sent to the control plane using a connection migration tool. In a sense, it lies somewhere between a solution to a SDN security issue and a SDN enhancement to network security. The solution to DoS in SDN is highlighted here while the traffic analysis and rule update functionality is discussed in Section V-B. Focusing on the Transport Control Protocol (TCP) SYN flood attack, AVANT-GUARD uses a connection migration tool to remove failed TCP sessions at the data plane prior to any notification to the control plane. This prevents the possibility of a saturation or DoS attack by only sending those flow requests to the control plane that complete the TCP handshake.

A replication component, CPRecovery, is presented in [56] as a means to provide network resilience. In the case of an attack from an external attacker that overwhelms the network operating system, the CPRecovery component provides seamless transition from the failed primary controller to a backup consistent with the network's failure-free state. The solution is limited to a centralized control implementation.

The source address validation scheme in [57] is a pre-emptive protection against IP spoofing, which could lead to a DoS attack. The scheme is called Virtual source Address Validation Edge (VAVE). The authors use both the traffic analysis capability of SDN and the ability to dynamically update rules to protect against IP spoofing. An incoming packet, which does not match against a rule at the OpenFlow switch is sent to the controller for source address validation. If spoofing is detected, the NOX-based OpenFlow controller installs a rule at the switch to stop traffic from that source address. A rule adaptor is used to limit the number of individual flow rules e.g. one rule can cover a flow space consisting of multiple source addresses. This, in turn, limits the capacity for a switch flow table flood attack.

Finally, the ident++ protocol presented in [58] could be implemented as a protection against DoS in that it avoids the central controller becoming a bottleneck in network management. The protocol queries end-hosts and networks on the path of a flow for additional information that the network administrator might not have so that users and end-hosts can participate in network security enforcement. This method of appropriate delegation of control and trust supports efficient network management.

The DoS solutions presented here point towards the use of SDN characteristics to overcome SDN attacks. With dynamic flow table management and distributed control, the threat of the SDN-specific DoS attack can actually be reduced. The potential to protect against any (traditional/SDN) network DoS attack using SDN will be discussed in Section V.

In Table VI, a summary of the problem/goal and the solution proposed for each research work presented under *Denial of Service* Issues in SDN is provided. [55], [56], [58] present different solutions to the problem of the controller bottleneck while [57] aims to protect against the data layer DoS attack.

D. Configuration Issues

Several solutions to the issue of network policy conflict arising from multiple applications in the SDN have been proposed in the literature [59]–[76], [84]–[86]. Different approaches have been taken but can be loosely grouped as those modelling network behaviour and using a custom algorithm to derive whether the network contains errors [59]–[62], real-time policy checking [63]–[67], [84], language-based approaches to deal with overlapping flow rules from different applications [69], [70] and a series of formal verification methods [73]–[76].

These solutions are discussed, by category, in the following sub-sections: *Detecting Network Errors*, *Real-Time Policy Checking*, *Language-Based Resolution*, *Consistent Abstractions/Network View*, and *Formal Verification Methods*.

1) Detecting Network Errors: NICE [59] combines model checking with symbolic execution to test OpenFlow applications for correctness. This solution enables the detection of when a network reaches an inconsistent network state. FlowChecker [60] uses binary decision diagrams to test for

intra-switch misconfigurations within a single flow table. Flover [61] uses assertion sets and modulo theories to verify flow policies. As each flow rule request is verified against the non-bypass properties enforced in the network, Flover supports a batch mode to improve the controller response time. Anteater [62] offers a static analysis approach to diagnose network configuration problems.

These solutions offer data plane verification. In general, they take between several seconds and a few hours to run and, as a result, may find problems after they occur rather than detecting issues in real-time to prevent potential damage to the network.

2) Real-Time Policy Checking: VeriFlow [63] studies the verification of invariants in real-time by intercepting flow rules before they reach the network. VeriFlow models the network as a graph to detect loops in the routing tables, unavailable paths etc. The objective is real-time network invariant detection and the results provided indicate performance in the order of hundreds of microseconds. NetPlumber [64] is also a real-time policy checking tool that incrementally checks for compliance of state changes. NetPlumber is an improvement on the author's previous Header Space Analysis (HSA) [84] work as it performs incremental computation based on the dependency subgraph so that it is fast enough to validate every update in real time. In [65], the authors apply HSA to develop a conflict detection and resolution algorithm for application in a SDN firewall. It is possible to bypass an SDN firewall by re-writing flow entries in switches. This solution creates and maintains a shifted flow graph based on checking the flow space and firewall authorization space for conflicting flow rules. The conflicting part of a flow path can then be blocked by inserting deny rules or by refusing a new flow rule. The implementation of the solution in the Floodlight firewall application provides a practical example of a policy conflict resolution mechanism, in contrast to many of the theoretical models. The limitation of the solution is that it only considers flow re-write actions related to firewall bypass threats.

Most recently, FlowGuard [66], [67] presents a solution to check network flow path spaces to detect firewall policy violations when network states are updated. FlowGuard also supports automatic and real-time violation resolution with a more refined approach than simply blocking/rejecting the policy update, as in some earlier solutions. This is achieved based on five different violation resolution strategies; flow rejecting, dependency breaking, update rejecting, flow removing and packet blocking. The strategy selected depends on the extent of the violation.

The focus on real-time operation of these solutions illustrates the evolution in policy conflict resolution techniques towards deployment in production networks. Further experimental deployments can be expected to bridge the gap between the theoretical models and the characteristics of live networks.

3) Language-Based Resolution: Frenetic [69] is a specific northbound API designed to resolve policy conflict. It is used for programming a collection of network switches controlled by a centralized controller. The run-time system converts flow rules into non-overlapping policies before instructing the controller to install the flow rules in the switches. In [70], the flow-based policy enforcement is implemented as a NOX application. Along with network isolation, it also allows the integration of external authentication sources to provide access control.

In contrast to the previous solutions, [69] and [70] introduce verification at the controller level presenting a policy conflict resolution mechanism that sits between the application layer and the control layer. By abstracting from the details of the low-level data plane rule implementation, this approach can simplify network programming to avoid policy conflicts. However, the current solutions require extension to operate in a distributed controller environment with the associated issues of distributed network state and multiple application resilience.

4) Consistent Abstractions/Network View: In [71], [85], the authors discuss abstractions for per-packet and per-flow consistency to overcome the instability of configuration changes in SDN. The per-packet consistency ensures that every packet in an update operation uses either the old policy or the new policy and not a combination of the two. The per-flow consistency is a generalization of per-packet consistency.

In [68], the authors propose a layered policy management solution. Intra- and inter-module flow rule dependencies are resolved at the application layer, inter-application dependencies at the control layer, and intra-table dependencies at the data layer. By tackling the overlapping flow rule issue at multiple layers, a wider set of dependencies can be resolved.

A concern with each of these proposed policy conflict resolution approaches is their scalability for large applications or large networks with regular flow rule additions/updates. In each case, the controller or a delegated control function at the switch performs extensive processing in order to detect potential network state misconfigurations. A potential alternative that does not sacrifice performance for the consistent network view is proposed in [72]. Rather than implementing a policy conflict resolution technique, the authors design a shared data store architecture. By maintaining a strongly consistent data store, the network state consistency is maintained.

5) Formal Verification Methods: Some formal approaches have also been proposed to determine if the SDN control-plane logic is safe, correct and secure overcoming errors in controller logic, conflicting application policies or incorrect assumptions about the operating environment [73]–[76].

Splendid Isolation [73] has been proposed as a means of verifying the isolation of program traffic across network slices. It is a formally proven programming abstraction for defining slices. FlowVisor [38], as previously introduced, is a hypervisor that acts as a transparent proxy between controllers and switches and enables an administrator to identify slices of the flowspace. However, it lacks the formal semantics and proof of correctness and security provided by Splendid Isolation. As described in [32], FlowVisor does not implement action isolation rendering the isolation mechanism vulnerable. For example, by altering the VLAN ID of a packet, a malicious controller could steal or inject packets into another network slice. Solutions such as [41], [73] are designed to overcome this issue.

Verificare [74] is a formal verification tool. Verificare incorporates specification modeling and verification of OpenFlow network correctness, convergence and mobility-related proper-

ties. In [75] the authors develop a detailed operational model of OpenFlow that they then formalize in the Coq proof assistant. Based on this model, they develop a verified compiler and runtime system and implement this as the first machine-verified SDN controller. VeriCon [76] verifies the safety of infinite state SDN programs.

It is clear from the range of contributions presented in this section that the issue of verifiable, consistent network state is a topic of research interest. The development of real-time network invariant detection tools and formal verification tools is valuable. However, the design of these tools for broad SDN deployments rather than OF-specific implementation is future research.

It should be noted that it is not just applications, but multiple entities that have write access to program the switch flow table, as shown in Figure 2. For example, as per the OpenFlow switch specification [36], the configuration points can determine which set of flows to program when bootstrapping the switch. If any application overrides or misconfigures these related flows without co-ordination, it could compromise the switch functionality. Similar problems can arise when network operators are given secure shell (ssh) access with write access to the switch flow table.

Configuration issues, such as this bootstrapping example, extend to the life-cycle of SDN networks and components, and directly impact on security. As the network changes over time with new applications, new devices, new users etc., security policies and technical methods must be implemented and maintained to ensure the protection of the network and its data. This is arguably more challenging for SDN than a traditional network as the open innovation model of SDN is likely to introduce more frequent network updates.

In Table VII, a summary of the problem/goal and the solution proposed for each research work presented under *Configuration Issues* in SDN is provided. The research work is split according to the five categories of solution identified in this Section. In general, *Detecting Network Errors* and *Language-Based Resolution* are early works in the field. *Real-Time Policy Checking* and *Consistent Abstractions/Network View* present further advances in policy conflict resolution, and *Formal Verification Methods* are some of the more recent contributions to this topic.

E. System Level SDN Security

In terms of securely implementing the SDN, a number of solutions have been presented.

In [77], a prototype network debugger is proposed to allow SDN developers to reconstruct the chain of events which lead to a bug and identify its root cause. This can support both network debugging and, for example, the auditing requirement highlighted in Section III for which the chain of events could provide useful network state information.

In [78], the authors present OFHIP, which is an integration of Host Identity Protocol (HIP) [87] and OpenFlow and uses IPSec encapsulating security payload (ESP). The objective of OFHIP is to enable secure mobility with OpenFlow by avoiding the problems that would be encountered with the current architecture given a change of IP address e.g. disrupted flow processing, active SSL/TLS session teardown, and mutual authentication issues. OFHIP proposes global identities as per HIP [87] introducing a cryptographic name space that is identical to the IPv6 address space, globally unique and does not change with mobility events. As a result, OFHIP enables OpenFlow switches to change their IP addresses securely supporting mobility within and between networks.

In [79], the authors extend the OFHIP work to consider security threats, issues and attacks in the Software-Defined Mobile Network control channel. A secure control channel architecture is presented and the protection of the architecture against a series of IP based attacks is analyzed. Results for protection against DoS, Replay, IP spoofing, and eavesdropping attacks demonstrate the potential for a secure software-defined mobile network implementation with the secure control channel design.

FRESCO [80] is one notable contribution, which presents an OpenFlow Security Application Development Framework incorporating FortNOX [52]; a security enforcement kernel described in Section IV-B. The idea behind FRESCO is to allow the rapid design and development of security specific modules, which can be incorporated as OpenFlow applications. A library of reusable modules for the detection and mitigation of network threats are provided.

The solutions presented for system level SDN security issues identify some of the diverse and dynamic environments in which SDN will be deployed e.g. cloud, data center, mobile. In order to provide secure visibility of network state across multiple, dynamic networks, novel security designs will be required.

In Table VIII, a summary of the problem/goal and the solution proposed for each research work presented under *System Level Security* Issues in SDN is provided. A broad range of solutions from network debugging to secure mobility can be identified.

V. Network Security Enhancement using the SDN Framework

As previously noted, there are clear benefits to be gained from the SDN architecture in terms of innovation in network usage. Six categories have been identified to capture the potential network security improvements based on the deployment of SDN. These categories are identified in Table IX and the individual solutions are discussed by category in the subsequent sections. The solutions described here take advantage of one or more of the SDN characteristics highlighted in Section II and Figure 2 to deliver improved network security. The clear observation from Table IX is the distinct lack of focus on the application-control interface in the research presented, while the other SDN layers/interfaces are quite evenly represented. It is most likely that this is due to the lack of a standardized protocol on this interface. In contrast, solutions can be proven and demonstrated using the OpenFlow protocol on the control-data interface. As a result, development of a secure application-control interface has been identified as a future research direction.

TABLE VII
PROBLEM AND SOLUTION PROPOSED FOR *Configuration Issues* IN SDN

Research Work	Problem/Goal	Proposed Solution
Detecting Network Errors		
NICE [59]	Test OF applications for correctness	Automated OF application testing to remove bugs in controllers
FlowChecker [60]	Avoid misconfiguration issues in OF due to conflicting flow rules	Use binary decision diagrams (BDDs) to test for intra-switch misconfigurations
Flover [61]	Verify that dynamically inserted flow policies do not violate the underlying network security policy	Use Satisfiability Modulo Theories (SMT) solver to detect if the aggregate of flow policies violates network security policy
Anteater [62]	Diagnose problems in the network data plane	Static analysis tool for checking invariants
Real-Time Policy Checking		
VeriFlow [63]	Real-time network invariant detection	Slice the OF network to check for invariant property violations
NetPlumber [64]	Verify network correctness in real time	Incremental computation to validate policy updates in real time
Security-Enhanced Firewall [65]	Detect and resolve firewall bypass threats in OF networks	Track flows using a shifted flow graph (HSA) and block conflicting flow path
FlowGuard [66], [67]	Detect and resolve firewall policy violations in dynamic OF network	Track network flow paths and check rule dependencies for automatic, real-time violation resolution
Language-Based Resolution		
Frenetic [69]	Resolve policy conflicts	Run-time system to convert flow rules into non-overlapping policies
Flow-Based Policy [70]	Simplify implementation of network security mechanisms in SDN	Flow-based network security policy language and framework
Consistent Abstractions/Network View		
Consistent Updates [71]	Overcome instability of configuration changes in SDN	Per-packet and per-flow consistency abstractions for configuration updates
LPM [68]	Manage complex network dynamics in SDN	Layered policy management framework (resolve inter-module, inter-application and intra-table dependencies)
Shared Data Store [72]	Maintain network performance while supporting a strongly consistent network view in SDN	Distributed, highly-available, strongly consistent controller for SDN based on fault-tolerant data store
Formal Verification Methods		
Splendid Isolation [73]	How to program shared networks in a secure and reliable manner?	Introduce slice-based network programming to isolate program traffic
Verificare [74]	A means to guarantee that SDN systems are safe, correct, or secure	Methodology and Tools for formally verifiable distributed system design
Machine-verified SDN [75]	Automatic checking of network-wide properties	Machine-verified SDN controller
VeriCon [76]	Formal method to prove the correctness of an SDN	Verification Tool for infinite-state SDN programs

A. Collect, Detect, Protect

A process of harvesting intelligence from existing Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), followed by analysis and centralized reprogramming of the network, can render the SDN more robust to malicious attacks than traditional networks. This process is well illustrated in [88]. The authors combine OpenFlow and sFlow [123] for anomaly detection and mitigation. Specifically, three modules are described in the solution architecture: (a) the Collector, in which flow statistics are gathered as is possible with the OpenFlow and sFlow protocols, (b) the Anomaly Detection, at which point analysis is carried out on the statistics and anomalies identified, and (c) the Anomaly Mitigation at which point flow-entries can be inserted in order to neutralize the identified attack. The combination of modules essentially acts as a feedback-control loop of network security. Successful detection of DDoS attacks, worm propagation and port scan attacks are demonstrated. Both the network-wide view and the dynamic programmability of the SDN are employed to successfully implement the solution in [88].

TABLE VIII
PROBLEM AND SOLUTION PROPOSED FOR *System Level Security* ISSUES IN SDN

Research Work	Problem/Goal	Proposed Solution
Debugger for SDN [77]	Simplify SDN debugging	Prototype network debugger for SDN
OFHIP [78]	Introduce Secure Mobility into OF switches and improve resilience against known TCP attacks	Global ID-based architecture enables OF switches to securely change IP address during mobility
Secure-SDMN [79]	Protect the Control Channel of Software-Defined Mobile Networks	Secure Control Channel Architecture based on IPSec tunnels and security gateways
FRESCO [80]	Simplify the development and deployment of complex security services for OF networks	Application Development Framework for Security Services Composition

TABLE IX
COMPARISON OF NETWORK SECURITY ENHANCEMENTS IN SDN

Security Enhancement	Research Work	SDN Layer/Interface				
		App	App-Ctl	Ctl	Ctl-Data	Data
Collect, Detect, Protect	Combining OpenFlow/sFlow [88], Active Security [89]	✓		✓	✓	✓
	Learning-IDS (L-IDS) [90], NetFuse [91], OrchSec [92]	✓		✓	✓	✓
	Cognition [93]	✓	✓	✓		
Traffic Analysis & Rule Updating	Resonance [94]	✓		✓	✓	✓
	AVANT-GUARD [55], Pedigree [95], OF-RHM [96]			✓	✓	✓
	SDN-MTD [97]	✓		✓	✓	✓
	NICE:NIDS [98], SnortFlow [99], SDNIPS [100], ScalableIDS [101]	✓		✓	✓	
	Revisiting Anomaly Detection [102]	✓		✓	✓	
	Fuzzy Logic SDN IDS [103]	✓		✓	✓	✓
DoS/DDoS Protection	Lightweight DDoS [104]	✓		✓	✓	
	CONA [105], DDoS Defender [106], DDoS Blocker [107]	✓		✓	✓	✓
Security Middleboxes - Architectures and Services	Slick [108], FlowTags [109]	✓	✓	✓	✓	✓
	SIMPLE-fying Middlebox [110]	✓		✓		✓
	OSTMA [111]			✓	✓	✓
	Covert Channel Protection [112]	✓		✓	✓	✓
	OpenSAFE [113], CloudWatcher [114]	✓	✓	✓	✓	
	Secure-TAS [115]				✓	✓
	Secure Forensics [116]			✓	✓	✓
AAA	AAA SDN [117]			✓	✓	✓
	C-BAS [118]	✓	✓	✓	✓	✓
Secure, Scalable Multi-Tenancy	vCNSMS [119], OpenvNMS [120], Tualatin [121]	✓		✓	✓	✓
	NetSecCloud [122]	✓		✓		

An illustration of this with respect to the SDN network diagram is provided in Figure 6. As previously noted, the statistics collection may be embedded in the controller or deployed as a separate function, as appropriate to the specific network.

Another complete feedback control loop methodology (collect, detect, protect) is that of *active security* introduced in [89]. An example illustrating intrusion detection, memory content capture and analysis, and network reconfiguration highlights the core capabilities of active security. These are defined as protect, sense, adjust, collect and counter. Once again, programmatic, centralized control enables this methodology.

In [90] a learning Intrusion Detection System (L-IDS) is proposed, which utilizes the SDN architecture to detect and respond to network attacks in embedded mobile devices. A learning switch controller is proposed to support routing correctness of mobile end-hosts. L-IDS takes advantage of the OpenFlow SDN architecture with IDS logic running in the controller and traffic measurement and anomaly detection processes built into the OpenFlow network devices. This supports efficient intrusion detection and dynamic network reconfiguration to mitigate attacks.

NetFuse [91] is a mechanism to protect against traffic overload in OpenFlow (OF)-based data center networks. NetFuse is an OF proxy device that monitors OF control messages and uses OF read state messages to build a picture of active network flows and resource utilization. A flow aggregation

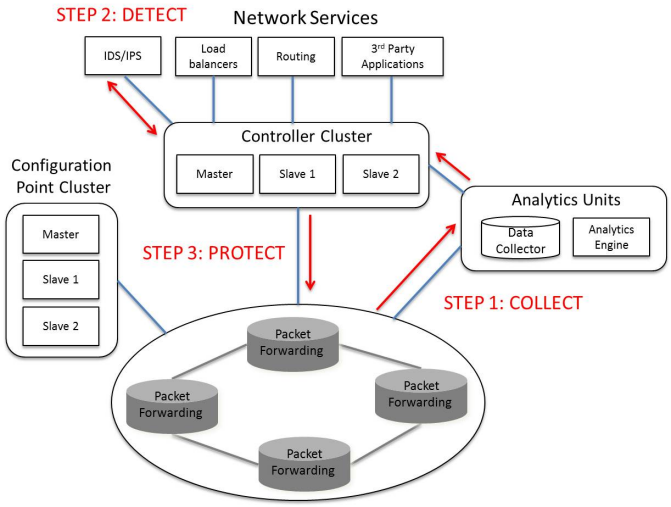


Fig. 6. SDN Security Feedback Control - Step 1: Collect network statistics, Step 2: Detect anomalies or intrusions in the network, Step 3: Insert flow rules to protect the network.

algorithm is then applied to identify the flows with overloading behaviour. A set of aggregation conditions are defined. For example, a volume of flows to a specific destination port may indicate an attack against specific services. The final step to limit the effect of a suspicious flow is adaptive rate-limiting based on a toxin-antitoxin mechanism; essentially the rate-limiting is adjusted based on the response of the overloading flow to the rate-limiting.

With OrchSec [92], an orchestrator-based architecture utilizing network monitoring and SDN control functions to develop security applications is proposed. The authors identify some of the deficiencies of other SDN-based security applications such as the performance impact of the overhead introduced at the SDN controller and the single point of failure when using only a single SDN controller. The OrchSec solution makes use of multiple diverse controller instances, uses sFlow for packet-level network monitoring, and develops applications that communicate via the northbound API (application-control interface) rather than being embedded in a controller. The functionality of OrchSec is demonstrated by detection and mitigation against a DNS amplification attack.

The concept of the feedback-control loop is extended in Cognition [93] with the proposal to apply cognitive functions at the SDN application plane. The features already available in open-source controllers to support enhanced network security such as global network view, switch statistics etc. (i.e. those features exploited in the work described earlier in this Section) are described as non-cognitive functions. The authors of Cognition propose to enhance the security level achieved with the non-cognitive functions through dynamic multi-objective optimization. The factors in the optimization problem would include environmental changes (e.g. intrusion/attack), network configuration changes, traffic changes and user changes. The idea is that the cognitive module uses learning-based approaches to anticipate a potential security threat and triggers the network to react appropriately to defend against the threat. The proposal is theoretical with experimental implementation identified as future work. This is a promising direction for future research.

A number of OpenFlow-based traffic monitoring solutions have been proposed that would support the initial *Collect, Detect* phases of the feedback control loop methodology [124]–[127]. OpenWatch [124] presents a prediction-based algorithm with which the granularity of flow measurement can be dynamically adjusted. FleXam [125] introduces a flexible sampling extension for OpenFlow so that specific packet-level information can be selected by the controller (or application). OpenSketch [126] is a software-defined traffic measurement architecture using sketches (compact data structures storing summary information about the state of packets) for customized and dynamic measurement data collection. In [127], the authors design and evaluate a hierarchical heavy hitters algorithm that identifies large traffic aggregates. Some of these solutions specifically identify their suitability for security applications although they can also be used in a wider context.

The solutions presented in this section demonstrate the great benefit to be gained from SDN for implementing reactive network security. Exciting further development can be expected along the lines of Cognition where the SDN flowspace is used for predictive functions and proactive network protection.

In Table X, a summary of the problem/goal and the solution proposed for each research work presented under *Collect, Detect, Protect* Network Security Enhancements in SDN is provided. A similar approach of statistics collection, analysis, and network reconfiguration is presented as a solution to three slightly different problems in [88]–[90]. In [91], rate-limiting is applied rather than network reconfiguration. [92] and [93] apply similar features to develop more secure SDN applications.

B. Attack Detection (Traffic Analysis) & Prevention (Rule Updating)

A separate category of SDN security enhancements are identified by their focus on a specific attack detection and, in some cases, prevention.

Resonance [94] was one of the first security-related SDN solutions. Resonance provides dynamic access control enforced by network devices based on higher-level security policies. Specifically, the high-level security policies are developed based on distributed network monitoring (traffic analysis), and enforced by programmable switches (rule updating). The system is proposed for securing enterprise networks.

AVANT-GUARD [55] proposes a solution to the control-data plane communication bottleneck in SDN by limiting the flow requests sent to the control plane using a connection migration tool. This was described in Section IV-C. In addition to the connection migration tool, an actuating trigger is also presented in [55]. This tool checks a defined traffic statistic condition against locally collected packet and flow statistics to determine an appropriate action. Two action options are presented, either to notify the control plane that the condition has been met or to proactively install a flow rule in a specified flow table. In the security context, the traffic statistic could be

TABLE X
PROBLEM AND SOLUTION PROPOSED FOR *Collect, Detect, Protect* NETWORK SECURITY ENHANCEMENTS IN SDN

Research Work	Problem/Goal	Proposed Solution
Combining OpenFlow/sFlow [88]	Avoid control plane overload (DoS) during OF statistics collection	SDN IDS/IPS based on Statistics Collection, Anomaly Detection and Anomaly Mitigation Modules
Active Security [89]	Dynamic and Programmable Security Infrastructure	Network Feedback Control providing integrated security
L-IDS [90]	Intrusion detection for embedded mobile devices	Use OF SDN to detect traffic anomalies and reconfigure network
NetFuse [91]	Prevent data center network overloading problems	OF proxy device to detect overload behaviour based on flow aggregation
OrchSec [92]	Overcome limitations of existing network security applications	Orchestrator-based SDN architecture to develop security applications
Cognition [93]	Enhance the security level of SDNs by applying cognitive functions at the app plane	A cognitive module implemented in the app plane (via dynamic multi-objective optimization)

the flow rate from individual hosts, the trigger could be a flow rate exceeding a defined threshold, and the action could be to install a flow rule to drop traffic from the identified host. AVANT-GUARD is shown to provide protection against network saturation attacks, network scanning attacks, and network intrusion attacks. However, only TCP flows are currently considered in the work.

The switch-controller communication of SDN is also exploited in [95] in a system called *Pedigree*. Tagstreams, which precede every connection, are sent to the controller. The controller checks the tag for an identifier corresponding to malware or to secret data. If such an identifier is detected, the controller installs a flow-rule at the switch to block the flow associated with this tagstream. Pedigree presents an interesting solution to stemming the spread of malware (including polymorphic worms) and preventing privilege escalation attacks. However, it is not clear that this solution with each tagstream sent to the controller is truly scalable.

Attackers use various scanning techniques to discover vulnerable targets in the network. Identifying the IP addresses in a target network is the first step in many attacks. In [96] the authors propose a solution called *OpenFlow Random Host Mutation* (OF-RHM) to prevent address discovery. An OpenFlow controller is used to manage a pool of virtual IP addresses, which are assigned to hosts within the network, hiding the real IP addresses from the outside world and presenting moving target defense. The real IP address remains unchanged internally while the externally-visible virtual IP addresses are randomly changed at regular intervals. The flow table size required to support OF-RHM increases with increasing network session establishment rate and session duration, which could be challenging for deployment of OF-RHM in a large network. DCPortals [128] provides traffic isolation for virtual networks in a virtualized data center by packet header rewriting to hide real traffic source and destination information. This guarantees that one tenant's traffic will never reach VMs of other tenants.

In SDN-MTD [97], the authors explore the use of SDN to provide network-based moving target defense (MTD) protection. There are a number of initial steps in an attack or an exploit including network mapping and reconnaissance, service discovery and operating system identification, and vulnerability detection. In SDN-MTD, the SDN framework enables attack surface obfuscation to delay and confuse the attacker. For example, to protect against network mapping, extra open and closed ports are revealed, and obfuscated port traffic is generated. This increases the cost of the attack to an attacker by requiring further investigation. The increased attack cost is measured for an Nmap [129] scan with the SDN-MTD protection implemented in Cisco's One Platform Kit (onePK) [130]. With SDN-MTD, the attack takes longer and more packets are transmitted. However, the network overhead to perform the filtering at the onePK controller is not considered.

NICE:NIDS [98] is an intrusion detection framework for virtual network systems. The use of virtual machines (VMs) in a cloud environment can introduce a range of vulnerabilities related to shared computing and storage resources and the potential for cloud users to install vulnerable software. NICE:NIDS uses an OF-based network intrusion detection agent to monitor and analyze network traffic. If a vulnerability is detected, the suspicious VMs can be quarantined and inspected, and ultimately protected based on a set of possible countermeasures such as traffic isolation, port blocking etc. The SDN characteristics support the NICE:NIDS dynamic, reconfigurable IDS, which could not be achieved in a traditional network.

With SnortFlow [99], the authors combine the intrusion detection capability of Snort [131] and the flexible network reconfiguration capability of OpenFlow. In the solution, the SnortFlow server component gathers data from Snort agents, performs the network security evaluation and generates the actions to be pushed to the controller based on which the controller reconfigures the network. The SnortFlow IPS is implemented in a Xen-based cloud environment. However, the results only illustrate the integration of Snort intrusion detection with the OF-based SDN and not an IPS. In a subsequent work, SDNIPS [100], the authors present evaluation

TABLE XI
PROBLEM AND SOLUTION PROPOSED FOR *Attack Detection (Traffic Analysis) & Prevention (Rule Updating)* NETWORK SECURITY ENHANCEMENTS IN SDN

Research Work	Problem/Goal	Proposed Solution
Resonance [94]	Improve enterprise network attack response capability	Dynamic access control system for securing enterprise networks
AVANT-GUARD [55]	Protect against Control Plane DoS attack and detect and respond to changing flow dynamics	Connection Migration Tool reducing data-control plane interaction and Actuating Trigger to install flow rules
Pedigree [95]	Defend enterprise networks against malware spread and data exfiltration	Traffic tainting (tagging) for flow tracking and filtering
OF-RHM [96]	Frequently change host IP addresses for moving target defense	Random Host Mutation using virtual-to-real IP translation
SDN-MTD [97]	Protect against network reconnaissance, service discovery and OS fingerprinting	SDN-based Moving Target Defense network protection application
NICE:NIDS [98]	Prevent vulnerable virtual machines in the cloud from being compromised	Network intrusion detection, measurement, and countermeasure selection mechanism
SnortFlow [99]	Overcome the latency, accuracy and flexibility issues of current IPS	OpenFlow-based IPS
SDNIPS [100]	A comprehensive IPS solution to reconfigure cloud networking on-the-fly	An SDN-based IPS solution
ScalableIDS [101]	Construct a scalable IDS to cope with increasing volume of network traffic	Scalable IDS architecture with sampling rate adjustment algorithm
Revisiting Anomaly Detection [102]	Use SDN to detect and contain home/home office network security problems	Anomaly Detection Algorithms deployed in NOX controller
Fuzzy Logic Sec. Mgmt. [103]	Use SDN to detect and protect the network from malicious attack	A fuzzy logic-based information security management system for SDN

results to demonstrate the increased attack detection rate of SDNIPS as compared to a traditional IPS implementation. This is based on the location of the detection engine in SDNIPS at Dom0 with direct access to monitor all tenant-based networks reducing processing time. Network reconfiguration (IPS) is also demonstrated with options to drop packets or redirect packets to a security appliance.

A similar proposal to integrate an open-source IDS with SDN is proposed in [101]. Suricata IDS [132] is used in this implementation with a simple sampling rate adjustment algorithm proposed to increase the scalability of the IDS architecture.

The value of using SDN to provide intrusion detection in a Home Office / Small Office environment is proposed in [102]. The implementation of four traffic anomaly detection algorithms in an SDN using OpenFlow and NOX (C++ based OpenFlow Controller [11]) is illustrated. The benefits in terms of efficient performance of the anomaly detection algorithms, and the ability to co-exist with other home network applications, such as Quality of Service (QoS) and Access Control, are identified. This solution exploits the standardized programmability of SDN to offer a more efficient anomaly detection service to the end-user than is provided by the traditional network deployment of anomaly detection in the network core.

A fuzzy-logic based IDS/IPS for SDN is presented in [103]. As in [102], the SDN characteristics enable efficient implementation of the popular anomaly detection algorithms. The use of fuzzy logic in decision-making reduces the computation requirements.

A range of IDS/IPS solutions are presented in this section to cope with different network attacks. Global network monitoring and programmability are key to delivering these solutions. However, in the majority of cases, further investigation/optimization of the solutions is required to deliver the performance and scalability necessary in full-scale SDN deployments.

In Table XI, a summary of the problem/goal and the solution proposed for each research work presented under *Attack Detection (Traffic Analysis) & Prevention (Rule Updating)* Network Security Enhancements in SDN is provided. The solutions range from dynamic access control to traffic tagging and filtering, and IPS to resolve both home office and enterprise network security problems.

C. DoS/DDoS Protection

The Denial-of-Service attack was identified in Section III as a weakness of SDN. However, a number of solutions exploit the combination of traffic analysis via the controller's network view, and the data plane programmability to produce efficient DoS/DDoS protection.

In [104], a Distributed DoS (DDoS) detection method is presented. The system monitors OpenFlow switches registered to a NOX controller at regular intervals to retrieve traffic data

TABLE XII
PROBLEM AND SOLUTION PROPOSED FOR *DoS/DDoS Protection* NETWORK SECURITY ENHANCEMENTS IN SDN

Research Work	Problem/Goal	Proposed Solution
Lightweight DDoS [104]	DDoS attack detection	Statistical information with self-organizing maps to classify traffic as normal or malicious
CONA [105]	DDoS attack detection and response in content-oriented network	Rate and pattern of content requests are analysed to detect DDoS attack
DDoS Defender [106]	DDoS attack detection and response	Use OF and LISP to detect and drop DDoS traffic based on traffic volume
DDoS Blocker [107]	Overcome difficulty of detecting and blocking DDoS attack by botnet	DDoS blocking scheme for SDN-managed network

such as average packets per flow, average bytes per flow etc. Self Organizing Maps, an artificial neural network, is then used to classify traffic as normal or malicious thus identifying abnormal flows.

A content-oriented networking architecture (CONA) is presented in [105]. An OpenFlow agent is used as an intermediary between content requests from a host to a content server in the network. The OpenFlow agent communicates with an OpenFlow controller. Content request messages are intercepted, analysed and filtered, as appropriate, in order to mitigate against DDoS attacks. A DDoS attack is identified when the rate of requests to a given content server exceeds a specified value. The controller is notified of the attack and sends rate limiting messages to each relevant CONA agent to limit attack propagation. It is not clear from the results provided how long it takes for the rate-limiting to be implemented and the attack to be halted.

A simple DDoS Defender is demonstrated in [106]. The authors use Locator/ID separation protocol (LISP) [133] to identify authorized and malicious sources. LISP separates the locator and identifier of the host so that only the locator changes as the node moves and the identifier remains unchanged. The DDoS attack is then detected based on traffic analysis. If the volume of traffic exceeds a set threshold, then the controller detects it and drops the packets.

Sophisticated DDoS attacks target specific services and with a large number of bots (compromised hosts) sending small volumes of legitimate-looking traffic, they can break through existing DDoS defenses. A DDoS Blocking application (DBA) is presented in [107] to overcome this type of botnet-based attack. DBA runs in the SDN controller monitoring flow metrics. If a DDoS attack is detected, legitimate traffic is redirected to a new server IP address while bots are blocked. The blocking mechanism relies on the inability of the bots to understand the redirect directive from the server so a computationally expensive machine-decoding format is required. The reliance of the DBA on the controller seeing each new flow may limit the wide deployment of this solution given that flow rules are commonly proactively installed rather than reactively.

As previously noted, the characteristics of SDN are well-suited to network DoS protection. There is also potential for application-level DoS defense. Future research to develop L4-7 security defenses with SDN is anticipated.

In Table XII, a summary of the problem/goal and the solution proposed for each research work presented under *DoS/DDoS Protection* Network Security Enhancements in SDN is provided. In each case, traffic statistics are used to detect the DDoS attack with the specific implementation dependent on the network environment.

D. Security Middleboxes - Architectures and Services

Middleboxes have long been associated with providing network security. Not surprisingly, hand-in-hand with the IDS work described in the previous sections is the integration of security middleboxes into SDN exploiting the benefit of programmability to redirect selected network traffic through the middlebox. Middleboxes serve both in analysing traffic for anomaly detection and in determining the appropriate action to mitigate against an identified attack. The advantage of IPS deployment based on SDN with unified scheduling of security applications across the whole network and load balancing among IPSs is highlighted in [134]. Solutions to the placement and integration of SDN middleboxes are considered in [108]–[112] while novel network security services deployed via SDN middleboxes are described in [113]–[116].

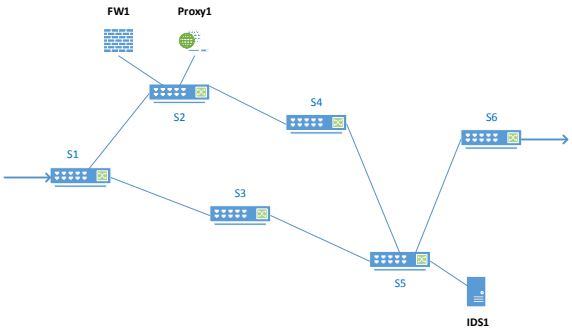
With the Slick architecture [108], the authors propose a centralized controller responsible for installing and migrating functions onto custom middleboxes. Applications direct the Slick controller to install functions for routing particular flows based on security needs. The benefit of this architecture is that new functions can be dynamically loaded to the Slick middleboxes providing flexibility. The Slick controller determines the placement of functions in the middleboxes and establishes the correct paths for specific traffic to pass through those functions removing the requirement to manually plan middlebox placement.

A similar architecture to Slick (augmented SDN controller and middlebox) is presented in [109]. However, a focus of the FlowTags architecture is a new Application Programming Interface (API) between the controller and FlowTags-enhanced middleboxes. The objective of FlowTags is to ensure consistent network policy enforcement as packet headers and contents may be dynamically modified by middleboxes. To ensure that the correct network policy chain can be applied to a flow (e.g. Firewall $->$ IDS $->$ Proxy), FlowTags are

embedded in packet headers to provide flow tracking and enable controlled routing of tagged packets. A low overhead is achieved with proactive installation of tagging rules in the switch flow tables. However, higher overhead is expected with reactive rule installation, which would be required in some real-world deployments.

In contrast to [108], [109], the SIMPLE policy enforcement layer [110] requires no modifications to SDN capabilities or middlebox functionality making it suitable for legacy systems. SIMPLE identifies the issue of ambiguity in forwarding decisions when using flow-based rules traditionally employed for L2/L3 functions (i.e. IP 5-tuple). The issue is illustrated in Fig. 7. To overcome this problem, SIMPLE attaches tags to packet headers to identify the processing state (i.e. location in the chain of switches and middleboxes along a policy chain) and tunnels packets between switches. In addition, load-balancing is performed across middleboxes. A tag-based packet classification architecture is also proposed in [135] to reduce filtering and flow management overhead. The additional processing time saved by the tag-based rather than hash-based classification is used for deep packet inspection.

In [111], an optimized security traversal with middlebox addition (OSTMA) mechanism is presented. The security traversal refers to the ordered sequence of passing through security devices/middleboxes and the objective is to meet the QoS guarantees of the network. The cost function is based on congestion in the network and a delay function based on middlebox occupancy/loading. OpenFlow SDN is employed to dynamically monitor and reconfigure the security traversal path. The more regularly the delay is measured by the OF controller, the fewer QoS violations that occur. However, the computation cost at the controller and the network reconfiguration communication cost must be considered to determine an appropriate optimization interval for a given network.



Policy Chain	Physical Sequence
FW1 – IDS1 – Proxy1	S1 S2 FW1 S2 S4 S5 IDS1 S5 S4 S2 Proxy1 S2 S4 S5 S6

Fig. 7. Example of data plane ambiguity to implement a policy chain: Firewall-IDS-Proxy. S5 sees the same packet three times and must choose between three actions: (1) Forward to IDS1, (2) Forward back to S2 for Proxy1, and (3) Send to the destination. [110]

In [112], the authors present a method of protection against covert channels. Covert channels are used for the secret transfer of information using means of communication not normally intended to be used for communication [136]. The solution in [112] uses the overview of the SDN controller along with SDN programmability to enable dynamic redirection of flows. Each node in the network has a set level of security clearance. The OpenFlow rules are set such that traffic is only authorized when the receiver security level is higher than the sender security level. If the receiver security level is lower than the sender security level then the data is passed through a filter (middlebox) in order to verify the security of the communication. The filter comprises a content check module and a time delay module. The content check module can operate at a range of levels; the higher the level, the greater the overhead. The most basic level involves checking TCP flag fields to discard illegitimate packets while the highest level looks for packet retransmission during the flow duration to identify a covert channel setup. The time delay module restricts covert timing channels by delaying the packet before forwarding it. The complete architecture is implemented using OpenFlow.

In terms of monitoring systems and access control methods deployed via SDN middleboxes, a number of solutions have been proposed. OpenSAFE [113] (Open Security Auditing and Flow Examination) uses its ALARMS policy language (A Language for Arbitrary Route Management for Security) to manage the routing of traffic through network monitoring devices. ALARMS describes paths between network elements and uses OpenFlow to interface to those network elements. For ALARMS constructions that are not natively understood by OpenFlow, the flow must be sent to the controller for processing. The authors identify this performance limitation of the solution. CloudWatcher [114] is an SDN application that controls network flows to guarantee that all necessary network packets are inspected by some security devices. It is designed for implementation in large and dynamic cloud networks where CloudWatcher uses the SDN characteristics of logically centralized control and programmability to automatically detour network packets to pre-installed network security devices. The example use-case is the ability of CloudWatcher to provide continuous security monitoring in the case of VM migration in a cloud environment.

A secure traffic analysis system (Secure-TAS) is presented in [115] to trace malicious activities on internal networks. SDN enables traffic redirection to/from suspicious hosts to the Secure-TAS. The system design consists of a transparent proxy, a threat analyser, a VM dispatcher, a victim host and an internet emulator to handle a wide range of attacks and to investigate malicious activities without the attacker being aware of the surveillance. The solution illustrates the potential for SDN to support advanced attack detection and prevention methods.

In [116], the authors explore the new opportunities for network forensics introduced by the SDN architecture. Specifically, two issues with forensic tools in traditional networks are identified; the inability to observe covert communication between compromised nodes, and the inability to detect attempts to exfiltrate sensitive data. It is proposed that these

TABLE XIII
PROBLEM AND SOLUTION PROPOSED FOR *Security Middlebox* NETWORK SECURITY ENHANCEMENTS IN SDN

Research Work	Problem/Goal	Proposed Solution
Slick [108]	Provide richer match/action set for improved network traffic management	Slick Controller and Middleboxes dynamically place network functions and direct traffic to those functions
FlowTags [109]	Ensure consistent network policy enforcement in the presence of middleboxes	Middleboxes add tags to outgoing packets to provide correct context
SIMPLE-fying Middlebox [110]	Efficient middlebox-specific traffic steering	Tag and tunnel packets between middleboxes
OSTMA [111]	Overcome the problem of QoS guarantee in security traversal	Dynamic security traversal scheme with middlebox addition for OF networks
Covert Channel Protection [112]	Restrict covert channels	Multi-level security network switch using OF filter
OpenSAFE [113]	Line-rate network traffic direction through security monitoring applications	Use OF to implement ALARMS policy for specifying and managing paths
CloudWatcher [114]	Provide monitoring services for cloud networks	SDN Application to control and direct network flows through security services
Secure-TAS [115]	Use SDN to protect the internal network from attack	Secure traffic analysis system to trace malicious activities on internal networks
Secure Forensics [116]	SDN-based forensic system to investigate faults including data exfiltration and collusion between compromised nodes	Lightweight middleboxes (Provenance Verification Points) to monitor and track network activity

issues can be overcome by using the network itself as an observation point. The solution is deployed with a set of SDN middleboxes called Provenance Verification Points (PVPs) that monitor network activity, authenticate message transmission, and provide a record of network transactions. The verification provided by the PVPs ensure the tracking necessary to detect the presence of network attacks.

While not related to middlebox implementation, two additional security policy enforcement solutions are worth mentioning [137], [138]. The SDN-based implementation of Amazon's EC2 Elastic IP and Security Groups [137] uses OF to provide flexibility to the provider and ease of configuration of policy preferences. This is achieved by bundling groups of EC2 instances (VMs) for security purposes e.g. a user could apply firewall rules to a group. LiveSec [138] is a security management architecture based on OF to apply fine-grained control on the end-to-end flows of network tenants or users. Once again, centralized control and network programmability enable the network configuration such that the appropriate security services are applied to the appropriate network flows.

The middlebox architecture and services research discussed in this section presents real novelty in network security provision. The ability to dynamically configure an attack protection based on real-time network behaviour is a first step. The potential to combine defenses and detect an attack without the awareness of the attacker(s) has great promise.

In Table XIII, a summary of the problem/goal and the solution proposed for each research work presented under *Security Middlebox* Network Security Enhancements in SDN is provided. The solutions presented in [108]–[112] are appliance-oriented architectures considering packet tagging and tunneling to apply security and network functions to traffic in the correct order. [113]–[116] introduce novel security services via SDN middleboxes.

E. Authentication, Authorization and Accounting

In Section IV, an authentication and access control mechanism was proposed as a solution to the issue of unauthorized access in SDN [51]. The ability of an OF-based SDN to support fine-grained access control to match services to identities was not highlighted in that work but could be considered as an SDN enhancement to network security.

An SDN-driven authentication, authorization, and accounting (AAA) system is presented in [117] to strengthen network security. The access control system is implemented by modifying the Floodlight [13] controller to register and authenticate switches, authenticate hosts and bind them to switches, to authenticate users, and to manage flows and user/host mobility. Authentication is supported by a RADIUS server. The advantage of this OF controller based access control system over a Kerberos [139] or Lightweight Directory Access Protocol (LDAP) [140] based system is not clearly defined. However, the fine-grained access control supported by SDN extends the AAA functionality.

The importance of efficient AAA management mechanisms for adoption of SDN experimentation facilities (EFs) is identified in [118]. The concern is raised based on the varying level of sophistication of AAA architectures in existing SDN EFs such as GENI [141] and OFELIA [142]. The authors propose a transferrable certificate-based AAA that can be implemented in any facility. The C-BAS architecture supports user identification via certificates, policing of user actions via credentials (role-based privileges), and full accountability. The

TABLE XIV
PROBLEM AND SOLUTION PROPOSED FOR *Authentication, Authorization, and Accounting (AAA)* NETWORK SECURITY ENHANCEMENTS IN SDN

Research Work	Problem/Goal	Proposed Solution
AAA-SDN [117]	Strengthen network security by SDN-driven access control	OF controller with authentication module
C-BAS [118]	Provide a robust, efficient AAA management mechanism for SDN experimental facilities (EFs)	A certificate-based AAA architecture for SDN EFs

solution presents an advance on existing AAA mechanisms specifically for SDN.

These two solutions illustrate the potential to increase AAA functionality and effectiveness by building on the SDN infrastructure. It is a clear example of using the network to protect the network. Further research on this topic is expected.

In Table XIV, a summary of the problem/goal and the solution proposed for each research work presented under *AAA* Network Security Enhancements in SDN is provided.

F. Secure, Scalable Multi-Tenancy

One of the characteristics of SDN identified in Section II is virtualized logical networks and the fact that SDN supports multi-tenancy. Several solutions have been proposed that exploit the characteristics of SDN to provide multi-tenant network security [119]–[122].

Some of the challenges encountered in multi-tenant virtualized networks include blurred network boundaries between virtual rather than physical systems, the correct placement of security devices to protect virtual logical boundaries, different security requirements for different tenants, and the dynamic nature of security policy based on VM migration. In [119], a collaborative network security prototype system (vCNSMS) for a multiple tenant datacenter network is described to overcome these challenges. SDN enables the network security system through efficient flow table management and the SDN controller network view that supports dynamic migration and reconfiguration to solve security policy inconsistencies.

Open virtual Network Management and Security (Open vNMS) [120] is proposed to support multi-tenancy while resolving network and Virtual Tenant Network (VTN) scalability issues. The flexible SDN architecture enables data link layer isolation overcoming the bottleneck of Virtual Local Area Network (VLAN), Multi-Protocol Label Switching (MPLS), and Generic Routing Encapsulation (GRE) encapsulation mechanisms that are limited by the number of available segments and the configuration complexity. Open vNMS is an OpenFlow v1.3 application taking advantage of multi-table pipeline processing. Tenant packets are processed by specific tables based on a slice ID. In the example implementation, an Open vNMS slice is a set of allocated and shared virtual OpenvSwitch resources (e.g. ports, tables, group tables). By this means, each tenant has control of its own isolated space.

Tualatin [121] also considers the requirement for network security protection in multi-tenant datacenters. Using SDN and OpenFlow, Tualatin offers fine-grained security protection for dynamically changing network topologies. The ability to implement fine-grained traffic classification with OpenFlow and apply individual security processing to selected traffic streams means that the system resource utilization can be optimized.

An example of SDN simplifying security in a cloud environment is presented in [122]. The dynamic requirements of cloud environments and the variety of software and operating systems under different individual and group control can present a challenge to the correct application of security policies. In [122], a security policy is based on data from system assessment, vulnerability assessment and live detection. This data is then used to generate a security score, a trust factor and a security requirement. An automated engine then implements this policy. This system is possible in SDN with a logically centralized database to provide up-to-date security information about each system or service hosted by the cloud provider. An evaluation environment is described in [122]. However, no results are provided and one concern would be system churn i.e. at what maximum frequency could this detection and reconfiguration engine be operated to avoid ping-pong re-configuration in the network? Is that maximum frequency reasonable with respect to normal cloud system operation?

The close relationship between SDN and NFV was identified in Section II. In [143], Zhang proposes a vision for cloud security that is underpinned by SDN. If individual Internet users can be provided with a "purer data" service (less spam, more secure), Zhang suggests that not only will there be an economic gain for society but there will also be less discontent. In his vision for cloud security, Internet users are provided secure Internet access by central dynamic provisioning of security services. SDN is the means to implement a software-defined nervous system for the network and a software-defined security system built into the network structure.

The solutions presented in this Section exploit the logically centralized control and programmability of the SDN architecture to provide secure, scalable multi-tenancy in both datacenter and cloud network environments. Specifically, fine-grained traffic classification enabled by OpenFlow and dynamic network reconfiguration provide the potential for service-targeted network security. The concept of Network Security as a Service is discussed in Section VIII.

In Table XV, a summary of the problem/goal and the solution proposed for each research work presented under *Secure, Scalable Multi-Tenancy* Network Security Enhancements in SDN is provided.

Based on the work presented in this section, it is clear that some of the SDN characteristics introduced in Section

TABLE XV
PROBLEM AND SOLUTION PROPOSED FOR *Secure, Scalable Multi-Tenancy* NETWORK SECURITY ENHANCEMENTS IN SDN

Research Work	Problem/Goal	Proposed Solution
vCNSMS [119]	Address network security in multiple tenant datacenter networks	A collaborative network security system with smart packet inspection using SDN
OpenvNMS [120]	Support multi-tenancy while resolving scalability issues	Autonomic SDN architecture supporting multi-tenancy and elastic isolation between tenant networks
Tualatin [121]	Provide network security services for tenant cloud infrastructures	Security Workload System for fine-grained dynamic network security protection
NetSecCloud [122]	Provide system/service-specific network security in cloud environments	Logically centralized database to provide latest system/service security information

II inherently simplify and improve the way network security is handled. Furthermore, the work illustrates the way in which novel solutions can change the security landscape of our infrastructure. The potential for simplified implementation of essential network security features offers a clear benefit to implementers deploying SDN-based networks.

VI. DISCUSSION: MORE OR LESS NETWORK SECURITY?

In the previous sections, Tables I, III and IX have been used to present a categorization of the research work to date on security in SDN.

Table I detailed the security analyses, which predominantly consider OpenFlow and vulnerabilities related to the implementation of an OF-based SDN. As a result, apart from [24], [27], the issues raised relate to the control and data plane layers. In addition, the question whether SDN introduces security vulnerabilities or enables improved network security was also discussed in several of the analyses. In order to address this discussion, the research work has been presented in terms of both solutions to security issues (Section IV and Table III) and security enhancements (Section V and Table IX). The wide range of research work suggests that SDN brings both improved functionality and open challenges, as summarized in this Section. It can also be noted from Tables III and IX that all layers and interfaces of SDN are strongly represented in the research solutions.

A. Improved Functionality

With respect to the SDN interfaces, the majority of the work surveyed references OpenFlow as the control-data interface. This is understandable given that OpenFlow has been a driving force for SDN and the benefit of demonstrating a solution using a standard protocol. In fact, starting from OpenFlow 1.3, the specification recommends several measures that can be used to mitigate the security threats of SDN. For example, rate-limiting to limit DoS attacks on the control plane, flow aggregation to encompass multiple flows under a single flow rule and thus prevent information disclosure, and shorter timeouts to reduce the impact of an attack that diverts traffic. These improvements to OpenFlow derive from lessons learned in early studies. Although any alternative to OpenFlow will likely be very similar in nature, it should be noted that OpenFlow may not be the only/definitive control-data interface protocol in SDNs.

Not surprisingly, without a standardized application-control interface (northbound API), the security enhancements discussed in Section V rarely refer to the means of communication between the application and controller. For the most part, if an application is proposed in the research work, it is written as a module of an existing controller. In contrast, a promising number of solutions discussed in Section IV consider the security requirements of the application-control interface. This interface presents a vulnerability to various attack vectors identified in Section III (e.g. Unauthorized Access, Malicious/Compromised Applications, and Configuration Issues). For this reason, further research is encouraged in this area in order to define solutions suitable for real-world deployment. The opportunity should also be taken at this early stage to build security into the design of a standard northbound API (or, indeed, any defined application-control interface).

B. Open Challenges

At face value, it would appear that an equal emphasis is being placed on the security enhancements and security issues in SDN. However, as previously noted, two of the security issues identified in Section II (Data Leakage and Data Modification) have not been considered in the literature. Of the issues that have been tackled, there is a concentration of focus around a number of key themes - unauthorized controller/application access, protection against DDoS attack, and resolution of network policy conflict arising from multiple applications programming the network. The independent contributions of the surveyed work are not yet mature enough for production deployment.

While it is positive to note the contribution to solutions to SDN security issues presented here, it remains a limited contribution as compared to the range of security issues identified in Table II. To achieve the full potential of SDN with multi-vendor interoperability, 3rd party network applications, and integration of network function virtualization, increased effort to tackle the challenges to network security introduced by the SDN framework is required.

In addition, from a deployment perspective, it is unclear how SDN can improve the operational security functions (e.g.

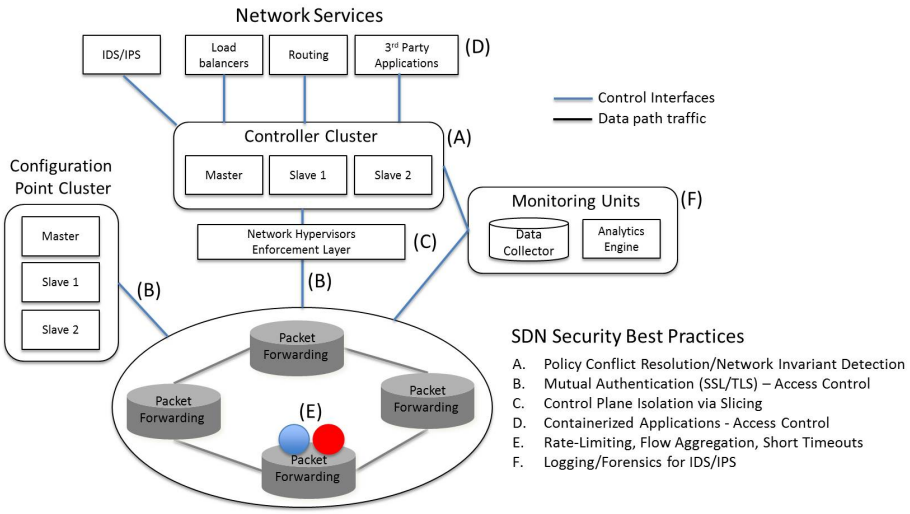


Fig. 8. SDN Security Recommended Best Practices

system reliability, non-stop forwarding, secure recovery from failures, audits etc.) in the production environment. There is minimal understanding as to how SDN-based solutions can replace traditional functions. To improve current security practices, a detailed assessment of how security is built, deployed and managed in real-world environments is required. Based on such an assessment, novel algorithms and effective functional solutions can be designed that can fundamentally change the dynamics of security in network infrastructures. In this regard, some potential future research directions in Section VIII are discussed.

C. Recommended Best Practices

In Figure 8, the **recommended best practices** that should be considered in an SDN deployment today are highlighted. The list reflects the SDN security solutions presented in this survey and lessons learned from existing SDN deployments.

- *Policy Conflict Resolution/Network Invariant Detection:* As exhibited by several conflict detection and resolution schemes, when application modules manipulate the network state, the controller should identify misconfigurations, unauthorized access, and irregularities to ensure correct functioning of the entities. Therefore, it is recommended that all controller solutions inherently include a policy conflict resolution subsystem to avoid network logic manipulation issues.
- *Mutual Authentication:* SDN components should support mutual authentication between all communicating entities. Authentication solutions should be enabled both within and across trust domains to avoid the introduction of insecure access to network resources. This prevents data manipulation attacks, impersonation of components and ensures secure identification of network entities
- *Control Plane Isolation via Slicing:* Unlike network virtualization, slicing the network resources will partition the resource allocated for tenants/users sharing the infrastructure. From a security standpoint, this provides an isolated environment that can be securely instantiated and protected from unauthorized access, data manipulations, and preventing data leakages. Isolation of resources can also guarantee dedicated resource allocation for individual tenants/users in the infrastructure.
- *Containerized Applications:* Assessing the different controller implementations, network applications are either statically compiled with the controller code, instantiated as a dynamic module with the controller software (e.g. in OpenDayLight the application bundles are dynamically loaded with the controller to exchange information and control the forwarding devices), or integrated as a third-party software with remote access to the controller. To prevent or restrict the impact of malicious application behavior, it is recommended to support application containerization, which can authenticate the application during setup, control the application's access rights on the infrastructure, and limit, account for and isolate the resource usage for each application. This can facilitate the secure introduction of third-party applications to the infrastructure and avoid any malicious behavior discussed in the previous sections.
- *Rate-Limiting, Flow Aggregation, Short Timeouts:* This recommended best practice highlights the correct use of inherent security features in a SDN framework. Specifically focusing on the data plane, the security of the forwarding element is dependent on both the correct use of configuration and control features and the security capabilities supported by the device. In OpenFlow protocol, the correct use of flow and switch attributes (e.g. flags, timeouts, mode of operation) as well as the inherent security features (e.g. metering to rate-limit the data flow to the control plane) can ensure correct packet forwarding behavior (i.e. avoid overlaps, notify flow deletes, operate securely when connection is lost with the controller etc.).
- *Logging/Forensics for IDS/IPS:* Network services and applications with monitoring capabilities require logging critical information and positively benefit from the logged

information when troubleshooting and debugging the infrastructure. As discussed in Section V, logging the network events can be valuable to network operations and can improve the security and reliability of the infrastructure.

VII. SDN Security in Industry: Open Standards and Open Source

While the research work presented here has been generated over the past 5 years, this interest has only been matched by industry working groups in the last 2 years. As SDN devices enter production and deployment, both the standardization industry and industry research groups have recognized the need to consider security in the SDN platform. The main groups considering standardization of networking related to SDN are are identified and their focus on security is highlighted.

A. ONF Security Discussion Group

The ONF Security Discussion Group [144] was launched in April 2013 and is currently working on defining essential and desired security functions for each element of the SDN platform. It became a full ONF Working Group in September 2014. The objectives of the Security group include: (1) the assessment of security considerations and related contribution to various ONF specifications (e.g. SDN Architecture 1.0, OpenFlow Table Type Patterns etc.) and standards (e.g. OpenFlow, OF-CONFIG etc.) [23], (2) to draft security standard documents (e.g. ONF Security Principles) with the objective of identifying and assessing the potential security risks associated with a reference SDN architecture, and (3) to encourage development of SDN applications that deliver network/data security.

It should be noted that the ONF Architecture document [145] also includes a section on security, which targets security of the control plane and the physical (or logical) devices, and security of the data traversing the network. The document describes the use of existing, proven, security models.

B. ETSI ISG NFV Security Expert Group

The ETSI Industry Specification Group (ISG) for Network Functions Virtualization (NFV) [146] takes a different approach to security. Rather than establishing a specific security working group, a set of security "experts" work across the NFV working groups.

The two stated goals of the NFV Security Expert Group are to design security into NFV from the beginning and to ensure security accreditation bodies address NFV. They have also clearly identified that they will not handle existing security issues, which are not altered by NFV.

Two group specifications have been issued [147], [148]. The security problem statement [147] identifies 10 key security issues. Each issue is assigned to a specific working group and proposed actions involve documentation of the existing solutions/recommended practice and subsequent research, if necessary. The security and trust guidance document [148] provides guidance for security considerations that are unique to NFV development, architecture and operation.

C. ITU-T Standardization of SDN

In December 2012 at the World Telecommunication Standardization Assembly (WTSA), a new resolution to expand and accelerate the ITU-T SDN standardization activity was agreed [149].

Two study groups (SGs) are tasked with the work, which was launched in June 2013. SG11 (Signalling requirements, protocols and test specifications) is tasked with developing the signalling requirements and protocols on SDN while SG13 (Future networks including cloud computing, mobile and NGN) is tasked with targeting the functional requirements and architectures of SDN [150]. The main output from the groups will be recommendations in each of these areas. A Joint Coordination Activity on SDN will coordinate the work.

The SG17 Security Group is considering SDN security under Q6/17. Regarding security *of* SDN (i.e. how to make SDN secure), they will link with SG13. Regarding security *by* SDN (i.e. how to provide security services using SDN principles), a new work item is being considered.

Two questions (Q7/13 and Q8/13) under study in SG13 refer to security in evolving networks and SDN; specifically Deep Packet Inspection (DPI), and security and identity management.

Other groups, which relate to the concept of SDN in terms of separation of forwarding and control planes, network configuration or routing are: IETF FORCES (Forwarding and Control Element Separation) [151], PCE (Path Computation Element) [152], ALTO (Application-Layer Traffic Optimization) [153], Netmod (NETCONF Data Modeling Language) [154], NETCONF (Network Configuration) [155], NVO3 (Network Virtualization Overlays) [156], LISP (Locator/ID Separation Protocol) [133] and I2RS (Interface to the Routing System) [157]. To the best of our knowledge, only I2RS makes specific reference to security requirements. The charter identifies the requirement to consider security in the high-level architecture and to react to network-based attacks. However, it can be noted that NetConf is required to run over a secure interface (secure socket shell (ssh) or TLS). In addition, the scope of NVO3 is such that security is inherent in its mandate; it requires traffic isolation, address independence and placement and migration of virtual machines. Finally, two IETF Internet-Draft documents were published in October 2012 entitled "Security Requirements in the Software Defined Networking Model" and "Security Analysis of the Open Networking Foundation (ONF) OpenFlow Switch Specification". However, neither document has been adopted by a working group to date.

The Software-Defined Networking Research Group (SDNRG) [158] of the Internet Research Task Force (IRTF) was chartered in January 2013. The group investigates SDN from a range of perspectives and aims to identify both near-term solutions and methods for SDN deployment, and longer-term research challenges [159]. One of the goals of SDNRG is to input to standards producing organizations such as the Internet Engineering Task Force (IETF), the European Telecommunications Standards Institute (ETSI) and the Open Networking Foundation (ONF). Security is a defined area of interest for

TABLE XVI
INDUSTRY FOCUS ON SECURITY IN SDN

Forum	Group Name	Launch Date	Objective	Proposed Output	Ref.
ONF	Security Working Group	Apr. 2013	Define security requirements for OpenFlow SDN architecture	SDN Security Standards Documents Threat Model/Analysis Document	[144]
ETSI	NFV Security Experts Group	Mar. 2013	Design Security into NFV from the start and ensure security accreditation bodies address NFV	Document existing solutions/ recommended practices and identify subsequent research requirements	[146]
ITU-T	Study Group SG11/SG13 (SG17)	Jun. 2013	Contribute to Standardization of SDN	Recommendations	[149], [150]

the group.

A summary of the industry standardization and research group's work regarding security in SDN is provided in Table XVI.

D. Open Source Activities

Many of the SDN developments are driven by open source implementations from academics and code contributors from industry. While there are several open source SDN solutions, those contributions related to security are highlighted here.

OpenFlowSec [160], a consortium of researchers from SRI International and Texas A&M University have built a SDN Security suite. Similar to a security enhanced Linux (SELinux), their suite extended the Floodlight controller to introduce SEFloodlight [50], which is an extension to their security enforcement kernel project (FortNOX [52]). In addition, the software suite includes a Security Actuator and OpenFlow Bothunter to invoke advanced security logic and to perform passive security analysis, respectively.

OpenDayLight provides AAA [161], a security module to authenticate identities, authorize administrative access to processes and applications, and provide accounting information to log accesses to resources. In addition, the project includes Defense4All [162] security project, which provides a system for detecting attack traffic and re-directing them based on OpenDayLight's monitoring and control capabilities. OpenDayLight also includes SNBI [163] a project used for securely bootstrapping the network infrastructure by automating the setup process for required devices and its credentials. For the SDN community to securely move forward, it is vital to build such solutions to improve the robustness of the software suite and enhance the introduction of novel security features.

Some of the security contributions to the open source activities derive from commercial security products developed to exploit the potential for enhanced and simplified security in SDNs. For example, RadWare DefenseFlow is the commercial product related to ODL Defense4All. Other SDN security products include Lancope StealthWatch System and HP Sentinel. Several of the offerings in the HP SDN App store [42] are also security-related providing DoS or Firewall protection.

VIII. FUTURE RESEARCH DIRECTIONS

A. Application-Network Transaction Security

A number of challenges are identified in Table II that affect the application layer and the application-control interface. Assuming that TLS is implemented and that mutual authentication takes place between the controller and network devices, a similar level of trust must be established between the applications and the network controller. Some initial solutions have been identified in Section IV. However, further extension of these solutions is possible.

For example, one of the means to advance resource sharing and customizability of the network is NFV [164]. The characteristics of SDN support NFV. Some examples of this have been described in Section V-F. SDN supports the provision of "X"-as-a-service (XaaS). This future service architecture will be driven by applications defining their required compute, storage and connection resources and requesting these from the network. This transaction must be secured. The key steps in such an exchange are illustrated in Fig. 9. First, the application sends a connection request to the controller from which the controller verifies the application identity and sends a connection response to the application. Next, the application verifies the controller identity and submits its requirement request (e.g. bandwidth, latency, counters to monitor etc.). At this point, the controller verifies the requirements, checking and resolving any potential policy conflicts arising from the request. Finally, the controller generates the necessary flow rules to implement the application service, updates neighbouring controllers of the changed network policy, and sends an acknowledgment to the application.

While some research solutions to provide secure application-network transactions have been proposed (as described in Section IV-B), the problem is far from resolved. It is important for both SDN Controller designers and NBI protocol developers to firmly consider security in their designs. A secure application-network transaction process would complement a potential secure control-data plane transaction (with mutual authentication) and secure control-control transactions to present a complete cross-layer secure SDN.

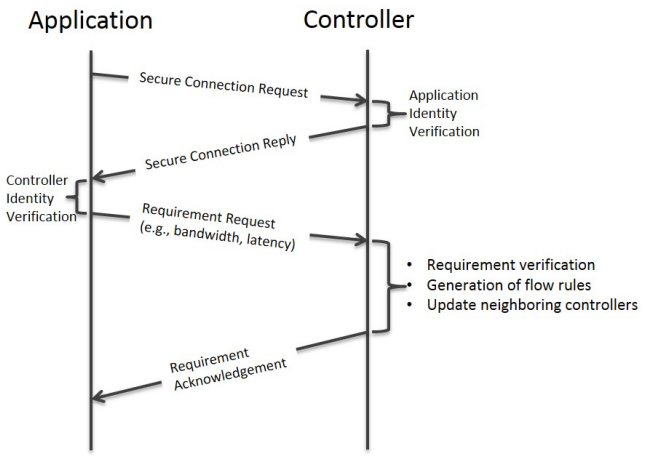


Fig. 9. Steps in the Secure Application-Network Transaction

B. Secure Network Map

As identified in Section III, the auditing process is a key concern in the wide area implementation of SDN particularly with the virtual to physical mappings of network elements and functions. It is important to be able to determine network state at any given time and to be able to track back to identify the network state at a previous point in time. SDN exists alongside virtualization and specifically NFV. This means that virtual networks are mapped onto physical networks for individual tenants. Network applications then draw on details of the underlying network in order to implement a given function e.g. load-balancing, energy management etc. The separation of sections of the network information base (NIB) may be possible to support multi-tenant isolation. However, the question of how to distribute or segment the NIB while providing the full network state information required by individual applications requires further study. The secure network map applies to the control and data layers and the control-data interface.

C. Exploiting SDN for Moving Target Defense

There is also great potential to exploit the adaptive and dynamic capabilities of the SDN framework in order to combat the types of attacker behavior presented by advanced persistent threats. The concept of security service insertion to dynamically detect and/or prevent suspicious traffic during live network operation has been discussed in Section V. Taking a step beyond that, there are opportunities for moving target defense to be applied in SDN, as in [96]. A clear candidate for moving target defense could be the controller. Rather than a single, central control element or distributed control across a series of elements, the control function could be shifted between a series of elements presenting a challenge to the attacker interested in targeting the control function. Derivation of potential algorithms to provide the shifting control function is another research topic. Study of moving target defense systems predominantly applies to the control layer.

D. Security Assessment Framework

SDN adopters (e.g. Service Providers, Enterprises etc.) are keen on deploying open-source solutions (e.g. OpenStack [165], OpenContrail [15], 3rd party network services etc.) to address their customer pain points. As this trend matures, it is evident that more open-source based network services/applications will be prominent in the market (e.g. HP App Store [42]). While network testbeds like Mininet [166], ProtoGENI [167] etc. can be used to test these components and assess their feature set and performance, an assessment framework to evaluate the component risks and vulnerabilities is still missing. Quagga SDN Module (QuaSM) has been proposed in [168] to support experimentation and testing of BGP functions within SDN with a view to support security researchers. It is proposed that information learned by BGP can be integrated with the rest of the SDN network state to provide increased network security. One potential research direction could involve building a framework to define security models and templates, known threat patterns etc. that could then be used to assess and verify the security of individual components. This would help adopters to understand the vulnerabilities prior to a potentially costly deployment. The development of a security assessment framework requires input at all layers and interfaces of the SDN.

E. Network Security as a Service (NSaaS)

Enterprises often lack the security expertise to protect their business functions. As a result they are dependent on external managed security service providers (MSSP) to secure their overall operations. Current MSSP solutions offer analytical capabilities, continuous resource monitoring, security device management, compliance evaluation etc. to prevent and detect vulnerabilities in the system. Most solutions involve manual assessment of resources and usage of sophisticated toolsets, which incurs high management costs for the enterprise. With the emergence of machine learning techniques (e.g. data correlation, feature extraction, graph mining etc.) and the capability to program the network with SDN, these MSSP functionalities can be automated and can reside closer to the enforcement point thereby offering an efficient alternative for enterprises. One potential research direction could involve building security-focused machine learning features, novel data-processing algorithms for security, and building SDN components to realize these functions in an automated manner. These features can be realized as virtualized network functions which are provisioned and managed by a SDN-based infrastructure. NSaaS involves all layers and interfaces of the SDN with specific focus on the application layer to provide novel security services.

F. Removing Middleboxes from the Network

In Section V, several proposals were discussed regarding the integration of network middleboxes. In each case, some improvement in middlebox management, functionality or flow tracking is suggested by use of the SDN characteristics. However, the requirement for the (potentially expensive) middlebox

itself is not questioned. In [169], the authors propose FAST (Flow-level State Transitions) as a new switch primitive. The concept of FAST is that a controller can proactively program state transitions that allow switches to run dynamic actions based on local information. For example, the implementation of a stateful firewall to filter unsolicited inbound TCP connections without any outbound flow would involve a TCP state machine to track the connection state and establish or close the connection dependent on the TCP state. In contrast, the reactive approach would require the switch to send TCP signals to the controller for TCP state tracking and subsequent flow rule installation with the associated latency overhead. By using the FAST process, an appropriate state machine can be built for any application with the potential for current switches to replace the variety of network functions generally supported by middleboxes. A potential research direction could be the development of this security state machine approach toward dynamic solutions suitable to protect against continuously evolving advanced persistent threats. The ability to call the state machine rather than deploy a new middlebox/middlebox software could provide the required speed of response. The removal of middleboxes from the network will focus on advances at the SDN data layer within network devices (physical and virtual).

IX. Conclusion

To respond to the question "Is SDN Secure?" at this stage, the response is most likely "Not really." It may be possible to have a secure SDN deployment if the SDN is deployed with equipment from a single provider, with no communication beyond the defined trust boundary, and in accordance with the strictest security principles. However, this deployment would only reflect a subset of the SDN characteristics and, as such, would be limited as compared to the full SDN potential.

In this survey, the evidence for the two sides of the SDN security coin has been presented; that it is possible to improve network security using the characteristics of the SDN architecture, and that the SDN architecture introduces security issues. The conclusion is that the work on enhancements to network security via SDN is more mature. This is evidenced by the commercially available applications.

However, research solutions have been presented to address some of the security issues introduced by SDN e.g. how to limit the potential damage from a malicious/compromised application. Work on these issues is developing encouraged by the increasing security focus of industry-sponsored standardization and research groups.

Having surveyed the research on security in SDN, a set of topics for future research have been identified. A strong theme amongst these topics is projection of potential security issues and automated response for quick reaction to network threats.

By implementing proven security techniques from our current network deployments, resolving known security issues in SDN, and further exploiting the dynamic, programmable, and open characteristics of SDN, software-defined networks may well be more secure than traditional networks. There is much work to do before this vision is realized.

References

[1] S. Horing, J. Menard, and R. Staehler, "Stored Program Controlled Network," *Bell System Technical Journal*, vol. 61, no. 7, 1982.

[2] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, 1997.

[3] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, "SANE: A protection architecture for enterprise networks," in *USENIX Security Symposium*, 2006.

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: Taking control of the enterprise," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 1–12.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[6] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, and M. Zhu, "B4: Experience with a globally-deployed software defined wan," in *Proceedings of the ACM SIGCOMM 2013 conference*. ACM, 2013, pp. 3–14.

[7] P. Patel, D. Bansal, L. Yuan, A. Murthy, A. Greenberg, D. A. Maltz, R. Kern, H. Kumar, M. Zikos, H. Wu, C. Kim, and N. Karri, "Ananta: Cloud Scale Load Balancing," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. ACM, 2013, pp. 207–218.

[8] S. Natarajan, A. Ramaiah, and M. Mathen, "A Software defined Cloud-Gateway automation system using OpenFlow," in *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*, Nov 2013, pp. 219–226.

[9] *VMware NSX Customer Story: Colt Decreases Data Center Networking Complexity*, VMware, Inc., 2014, http://blogs.vmware.com/networkvirtualization/2014/08/vmware-nsx-customer-story-colt-decreases-data-center-networking-complexity.html.

[10] *Software Defined Networking: Gaining Momentum*, Nuage Networks, 2014, http://www.nuagenetworks.net/momentum/.

[11] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[12] D. Erickson, "The beacon openflow controller," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 13–18.

[13] "Floodlight Controller, Floodlight Documentation, For Developers, Architecture." [Online]. Available: http://www.projectfloodlight.org/floodlight/

[14] "OpenDaylight: A Linux Foundation Collaborative Project," 2014. [Online]. Available: http://www.opendaylight.org

[15] "OpenContrail." [Online]. Available: http://opencontrail.org/

[16] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A Distributed Control Platform for Large-scale Production Networks." in *OSDI*, vol. 10, 2010, pp. 1–6.

[17] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and flexible cellular core network architecture," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 163–174.

[18] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*. USENIX Association, 2010, pp. 3–3.

[19] S. H. Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 19–24.

[20] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, and W. Snow, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[21] D. Kreutz, F. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *arXiv preprint arXiv:1406.0440*, 2014.

[22] M. Jarschel, T. Zinner, T. Hofeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for sdn," *Communications Magazine, IEEE*, vol. 52, no. 6, pp. 210–217, 2014.

[23] "ONF Specifications." [Online]. Available: https://www.opennetworking.org/sdn-resources/technical-library

[24] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN Security: A Survey," in *IEEE SDN for Future Networks and Services (SDN4FNS)*, 2013, pp. 1–7.

[25] R. Kloeti, "OpenFlow: A Security Analysis," April 2013. [Online]. Available: ftp://yosemite.ee.ethz.ch/pub/students/2012-HS/MA-2012-20_signed.pdf

[26] K. Benton, L. J. Camp, and C. Small, "OpenFlow Vulnerability Assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 151–152.

[27] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 55–60.

[28] D. Li, X. Hong, and J. Bowman, "Evaluation of Security Vulnerabilities by Using ProtoGENI as a Launchpad," in *Global Telecommunications Conference (GLOBECOM 2011)*. IEEE, 2011, pp. 1–6.

[29] S. Shin and G. Gu, "Attacking Software-Defined Networks: The First Feasibility Study," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 165–166.

[30] R. Smeliansky, "SDN for network security," in *Science and Technology Conference (Modern Networking Technologies)(MoNeTeC), 2014 First International*. IEEE, 2014, pp. 1–5.

[31] L. Schehlmann, S. Abt, and H. Baier, "Blessing or curse? Revisiting security aspects of Software-Defined Networking," in *Network and Service Management (CNSM), 2014 10th International Conference on*. IEEE, 2014, pp. 382–387.

[32] V. T. Costa and L. H. M. K. Costa, "Vulnerability Study of FlowVisor-based Virtualized Network Environments," in *2nd Workshop on Network Virtualization and Intelligence for the Future Internet*, 2013.

[33] A. Y. Ding, J. Crowcroft, S. Tarkoma, and H. Flinck, "Software defined networking for security enhancement in wireless mobile networks," *Computer Networks*, vol. 66, pp. 94–101, 2014.

[34] M. Tsugawa, A. Matsunaga, and J. A. Fortes, *Cloud Computing Security: What Changes with Software-Defined Networking?*, ser. Secure Cloud Computing. Springer, 2014, pp. 77–93.

[35] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, "Threat modeling-uncover security design flaws using the stride approach," *MSDN Magazine-Louisville*, pp. 68–75, 2006.

[36] "OpenFlow Switch Specification Version 1.4," Open Networking Foundation. [Online]. Available: https://www.opennetworking.org

[37] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, ser. CCS '09. ACM, 2009, pp. 199–212.

[38] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Flowvisor: A network virtualization layer," *OpenFlow Switch Consortium, Tech.Rep*, 2009.

[39] A. Al-Shabibi, M. D. Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "OpenVirteX: make your virtual SDNs programmable," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 25–30.

[40] "OpenVirtex (OVX) Network Hypervisor." [Online]. Available: www.openvirtex.org

[41] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *Internet Computing, IEEE*, vol. 17, no. 2, pp. 20–27, 2013.

[42] *SDN Dev Center: Unlock Network Innovation*, Hewlett Packard Company, 2014, www.hp.com/go/sdndevcenter.

[43] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for SDN? Implementation challenges for software-defined networks," *Communications Magazine, IEEE*, vol. 51, no. 7, 2013.

[44] O. O. MM and K. OKAMURA, "Securing Distributed Control of Software Defined Networks," *International Journal of Computer Science & Network Security*, vol. 13, no. 9, 2013.

[45] H. Li, P. Li, S. Guo, and S. Yu, "Byzantine-resilient secure software-defined networks with multiple controllers," in *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 695–700.

[46] D. Yu, A. W. Moore, C. Hall, and R. Anderson, *Authentication for Resilience: The Case of SDN*, ser. Security Protocols XXI. Springer, 2013, pp. 39–44.

[47] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, "Towards a secure controller platform for openflow applications," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 171–172.

[48] S. Scott-Hayward, C. Kane, and S. Sezer, "OperationCheckpoint: SDN Application Control," in *22nd IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2014, pp. 618–623.

[49] OPENFLOWSEC.ORG, "Security-Enhanced Floodlight." [Online]. Available: www.openflowsec.org

[50] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, "Securing the Software-Defined Network Control Layer," in *Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS)*, February 2015.

[51] D. M. F. Mattos, L. H. G. Ferraz, and O. C. M. B. Duarte, "AuthFlow: Authentication and Access Control Mechanism for Software Defined Networking."

[52] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 121–126.

[53] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A Robust, Secure, and High-Performance Network Operating System," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 78–89.

[54] B. Chandrasekaran and T. Benson, "Tolerating SDN application failures with LegoSDN," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. ACM, 2014, p. 22.

[55] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security*. ACM, 2013, pp. 413–424.

[56] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, "A replication component for resilient OpenFlow-based networking," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 933–939.

[57] G. Yao, J. Bi, and P. Xiao, "Source address validation solution with OpenFlow/NOX architecture," in *19th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2011, pp. 7–12.

[58] J. Naous, R. Stutsman, D. Mazieres, N. McKeown, and N. Zeldovich, "Delegating network security with more information," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 19–26.

[59] M. Canini, D. Venzano, P. Peresini, D. Kostic, and J. Rexford, "A NICE way to test OpenFlow applications," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2012.

[60] E. Al-Shaer and S. Al-Haj, "FlowChecker: Configuration analysis and verification of federated OpenFlow infrastructures," in *Proceedings of the 3rd ACM workshop on Assurable and usable security configuration*. ACM, 2010, pp. 37–44.

[61] S. Son, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Model Checking Invariant Security Properties in OpenFlow." [Online]. Available: http://faculty.cse.tamu.edu/guofei/paper/Flover-ICC13.pdf

[62] H. Mai, A. Khurshid, R. Agarwal, M. Caesar, P. Godfrey, and S. T. King, "Debugging the data plane with anteater," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 290–301, 2011.

[63] A. Khurshid, W. Zhou, M. Caesar, and P. Godfrey, "VeriFlow: Verifying network-wide invariants in real time," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 467–472, 2012.

[64] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real time network policy checking using header space analysis," in *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.

[65] J. Wang, Y. Wang, H. Hu, Q. Sun, H. Shi, and L. Zeng, *Towards a Security-Enhanced Firewall Application for OpenFlow Networks*, ser. Cyberspace Safety and Security. Springer, 2013, pp. 92–103.

[66] H. Hu, G.-J. Ahn, W. Han, and Z. Zhao, "Towards a Reliable SDN Firewall," *Presented as part of the Open Networking Summit 2014 (ONS 2014)*, 2014.

[67] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "FLOWGUARD: Building Robust Firewalls for Software-Defined Networks," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 97–102.

[68] W. Han, H. Hu, and G.-J. Ahn, *LPM: Layered Policy Management for Software-Defined Networks*, ser. Data and Applications Security and Privacy XXVIII. Springer, 2014, pp. 356–363.

[69] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: A network programming language," *ACM SIGPLAN Notices*, vol. 46, no. 9, pp. 279–291, 2011.

[70] T. Hinrichs, N. Gude, M. Casado, J. Mitchell, and S. Shenker, "Expressing and enforcing flow-based network security policies," *University of Chicago, Tech.Rep*, 2008.

[71] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: Change you can believe in!" in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. ACM, 2011, p. 7.

[72] F. A. Botelho, F. M. V. Ramos, D. Kreutz, and A. N. Bessani, "On the feasibility of a consistent and fault-tolerant data store for SDNs," in *Software Defined Networks (EWSDN), 2013 Second European Workshop on*. IEEE, 2013, pp. 38–43.

[73] C. Schlesinger, A. Story, S. Gutz, N. Foster, and D. Walker, "Splendid Isolation: Language-Based Security for Software-Defined Networks," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 79–84.

[74] R. W. Skowyra, A. Lapets, A. Bestavros, and A. Kfoury, "Verifiably-safe software-defined networks for CPS," in *Proceedings of the 2nd ACM international conference on High confidence networked systems*. ACM, 2013, pp. 101–110.

[75] A. Guha, M. Reitblatt, and N. Foster, "Machine-verified network controllers," in *ACM SIGPLAN Notices*, vol. 48. ACM, 2013, pp. 483–494.

[76] T. Ball, N. Bjrner, A. Gember, S. Itzhaky, A. Karbyshev, M. Sagiv, M. Schapira, and A. Valadarsky, "VeriCon: towards verifying controller programs in software-defined networks," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2014, p. 31.

[77] N. Handigol, B. Heller, V. Jeyakumar, D. Mazieres, and N. McKeown, "Where is the debugger for my software-defined network?" in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 55–60.

[78] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, "Enabling Secure Mobility with OpenFlow," in *IEEE Software Defined Networks for Future Networks and Services*. IEEE, 2013.

[79] M. Liyanage, M. Ylianttila, and A. Gurtov, "Securing the control channel of software-defined mobile networks," pp. 1–6, 2014.

[80] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyso, "FRESCO: Modular composable security services for software-defined networks," in *Proceedings of Network and Distributed Security Symposium*, 2013.

[81] J. McCauley, A. Panda, M. Casado, T. Koponen, and S. Shenker, "Extending SDN to large-scale networks," *Open Networking Summit*, pp. 1–2, 2013.

[82] S. Scott-Hayward, "Design and deployment of secure, robust and resilient sdn controllers," in *Proceedings of the 2015 IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–5.

[83] F. Botelho, A. Bessani, F. M. Ramos, and P. Ferreira, "On the design of practical fault-tolerant sdn controllers," in *Proc. of the 3rd European Workshop on Software Defined NetworksEWSDN*, vol. 14, 2014.

[84] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks." in *NSDI*, 2012, pp. 113–126.

[85] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*. ACM, 2012, pp. 323–334.

[86] S. Natarajan, X. Huang, and T. Wolf, "Efficient conflict detection in flow-based virtualized networks," in *Computing, Networking and Communications (ICNC), 2012 International Conference on*, Jan 2012, pp. 690–696.

[87] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson, "Host identity protocol," *RFC5201, April*, 2008.

[88] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable Anomaly Detection and Mitigation mechanism on SDN Environments," *Computer Networks*, 2013.

[89] R. Hand, M. Ton, and E. Keller, "Active Security," in *ACM SIGCOMM Hot Topics in Networks*, 2013.

[90] R. Skowyra, S. Bahargam, and A. Bestavros, "Software-Defined IDS for Securing Embedded Mobile Devices," 2013. [Online]. Available: http://www.cs.bu.edu/techreports/pdf/2013-005-software-defined-ids.pdf

[91] Y. Wang, Y. Zhang, V. Singh, C. Lumezanu, and G. Jiang, "NetFuse: Short-circuiting traffic surges in the cloud," in *2013 IEEE International Conference on Communications (ICC)*. IEEE, 2013, pp. 3514–3518.

[92] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "OrchSec: An orchestrator-based architecture for enhancing network-security using Network Monitoring and SDN Control functions," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–9.

[93] E. Tantar, M. R. Palattella, T. Avanesov, M. Kantor, and T. Engel, *Cognition: A Tool for Reinforcing Security in Software Defined Networks*, ser. EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V. Springer, 2014, pp. 61–78.

[94] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: dynamic access control for enterprise networks," in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 11–18.

[95] A. Ramachandran, Y. Mundada, M. B. Tariq, and N. Feamster, "Securing enterprise networks using traffic tainting," 2009.

[96] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 127–132.

[97] P. Kampanakis, H. Perros, and T. Beyene, "SDN-based solutions for Moving Target Defense network protection," in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*. IEEE, 2014, pp. 1–6.

[98] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2013.

[99] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar, "Snortflow: A openflow-based intrusion prevention system in cloud environment," in *Research and Educational Experiment Workshop (GREE), 2013 Second GENI*. IEEE, 2013, pp. 89–92.

[100] T. Xing, Z. Xiong, D. Huang, and D. Medhi, "SDNIPS: Enabling Software-Defined Networking Based Intrusion Prevention System in Clouds," pp. 308–311, 2014.

[101] C. Jeong, T. Ha, J. Narantuya, H. Lim, and J. Kim, "Scalable network intrusion detection on virtual SDN environment," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 264–265.

[102] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*. Springer, 2011, pp. 161–180.

[103] S. Dotcenko, A. Vladyko, and I. Letenko, "A fuzzy logic-based information security management for software-defined networks," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 2014, pp. 167–171.

[104] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE 35th Conference on Local Computer Networks (LCN)*. IEEE, 2010, pp. 408–415.

[105] J. Suh, H. Choi, W. Yoon, T. You, T. Kwon, and Y. Choi, "Implementation of Content-oriented Networking Architecture (CONA): A Focus on DDoS Countermeasure," in *European NetFPGA Developers Workshop*, 2010.

[106] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen, "A novel design for future on-demand service and security," in *Communication Technology (ICCT), 2010 12th IEEE International Conference on*. IEEE, 2010, pp. 385–388.

[107] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, "A SDN-oriented DDoS blocking scheme for botnet-based attacks," in *Ubiquitous and Future Networks (ICUFN), 2014 Sixth International Conf on*. IEEE, 2014, pp. 63–68.

[108] B. Anwer, T. Benson, N. Feamster, D. Levin, and J. Rexford, "A slick control plane for network middleboxes," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 147–148.

[109] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *Proc. NSDI*, 2014.

[110] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying Middlebox Policy Enforcement Using SDN." ACM SIGCOMM, August 2013.

[111] Y.-J. Chen, F.-Y. Lin, and L.-C. Wang, "Dynamic Security Traversal in OpenFlow Networks with QoS Guarantee," *International Journal of Science and Engineering*, vol. 4, no. 2, pp. 251–256, 2014.

[112] X. Liu, H. Xue, X. Feng, and Y. Dai, "Design of the multi-level security network switch system which restricts covert channel," in *IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2011, pp. 233–237.

[113] J. R. Ballard, I. Rae, and A. Akella, "Extensible and scalable network monitoring using OpenSAFE," *Proc.INM/WREN*, 2010.

[114] S. Shin and G. Gu, "CloudWatcher: Network security monitoring using OpenFlow in dynamic cloud networks (or: How to provide security monitoring as a service in clouds?)," in *20th IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2012, pp. 1–6.

[115] S. Hirono, Y. Yamaguchi, H. Shimada, and H. Takakura, "Development of a secure traffic analysis system to trace malicious activities on internal networks," in *Computer Software and Applications Conference (COMPSAC), 2014 IEEE 38th Annual*. IEEE, 2014, pp. 305–310.

[116] A. Bates, K. Butler, A. Haeberlen, M. Sherr, and W. Zhou, "Let SDN Be Your Eyes: Secure Forensics in Data Center Networks," in *Workshop on Security of Emerging Networking Technologies (SENT)*, 2014.

[117] V. Dangovas and F. Kuliesius, "SDN-Driven Authentication and Access Control System," in *The International Conference on Digital Information, Networking, and Wireless Communications (DINWC2014)*. The Society of Digital Information and Wireless Communication, 2014, pp. 20–23.

[118] U. Toseef, A. Zaalouk, T. Rothe, M. Broadbent, and K. Pentikousis, "C-BAS: Certificate-based AAA for SDN experimental facilities," in *2014 Third European Workshop on Software Defined Networks (EWSDN)*. IEEE, 2014, pp. 91–96.

[119] Z. Chen, W. Dong, H. Li, J. Cao, P. Zhang, and X. Chen, "Collaborative Network Security in Multi-Tenant Data Center for Cloud Computing," *Tsinghua Science and Technology*, vol. 1, p. 009, 2014.

[120] M. F. Ahmed, C. Talhi, M. Pourzandi, and M. Cheriet, "A Software-Defined Scalable and Autonomous Architecture for Multi-tenancy," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 568–573.

[121] X. Wang, Z. Liu, J. Li, B. Yang, and Y. Qi, "Tualatin: Towards network security service provision in cloud datacenters," in *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*. IEEE, 2014, pp. 1–8.

[122] S. Seeber and G. D. Rodosek, "Improving Network Security Through SDN in Cloud Scenarios," pp. 376–381, 2014.

[123] "sFlow - Sampling Technology for Network Traffic Monitoring." [Online]. Available: www.sflow.org

[124] Y. Zhang, "An adaptive flow counting method for anomaly detection in SDN," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 25–30.

[125] S. Shirali-Shahreza and Y. Ganjali, "Efficient Implementation of Security Applications in OpenFlow Controller with FleXam," in *High-Performance Interconnects (HOTI), 2013 IEEE 21st Annual Symposium on*. IEEE, 2013, pp. 49–54.

[126] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with opensketch," in *Proceedings 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI*, vol. 13, 2013.

[127] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches," in *Proc. of the USENIX HotICE workshop*, 2011.

[128] R. V. Nunes, R. L. Pontes, and D. Guedes, "Virtualized network isolation using software defined networks," in *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*. IEEE, 2013, pp. 683–686.

[129] "Nmap." [Online]. Available: http://nmap.org/

[130] "Cisco onePK." [Online]. Available: http://www.cisco.com/c/en/us/products/ios-nx-os-software/onepk.html

[131] "Snort - Open Source Intrusion Prevention System." [Online]. Available: https://www.snort.org

[132] "Open Source Intrusion Detection and Prevention System." [Online]. Available: http://suricata-ids.org

[133] IETF LISP (Locator/ID Separation Protocol). [Online]. Available: http://datatracker.ietf.org/wg/lisp/

[134] L. Zhang, G. Shou, Y. Hu, and Z. Guo, "Deployment of Intrusion Prevention System based on Software Defined Networking," in *Communication Technology (ICCT), 2013 15th IEEE International Conference on*. IEEE, 2013, pp. 26–31.

[135] H. Farhadi and A. Nakao, "Rethinking Flow Classification in SDN," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 598–603.

[136] S. Zander, G. J. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," *IEEE Communications Surveys and Tutorials*, vol. 9, no. 1-4, pp. 44–57, 2007.

[137] G. Stabler, A. Rosen, S. Goasguen, and K.-C. Wang, "Elastic IP and security groups implementation using OpenFlow," in *Proceedings of the 6th international workshop on Virtualization Technologies in Distributed Computing Date*. ACM, 2012, pp. 53–60.

[138] K. Wang, Y. Qi, B. Yang, Y. Xue, and J. Li, "LiveSec: Towards effective security management in large-scale production networks," in *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*. IEEE, 2012, pp. 451–460.

[139] B. C. Neuman and T. Tsó, "Kerberos: An authentication service for computer networks," *Communications Magazine, IEEE*, vol. 32, no. 9, pp. 33–38, 1994.

[140] "Lightweight directory access protocol (LDAP): The protocol," *IETF RFC 4511*, 2006.

[141] "GENI: Global Environment for Network Innovation." [Online]. Available: http://www.geni.net

[142] "OFELIA: OpenFlow in Europe - Linking Infrastructure and Applications." [Online]. Available: www.fp7-ofelia.eu

[143] H. Zhang, "A vision for cloud security," *Network Security*, vol. 2014, no. 2, pp. 12–15, 2014.

[144] Open Networking Foundation Security Working Group. [Online]. Available: https://www.opennetworking.org/technical-communities/areas/services

[145] "SDN Architecture (Issue 1)," June, 2014. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[146] ETSI ISG Network Functions Virtualization Security Expert Group. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv

[147] "Network Functions Virtualization (NFV) - NFV Security - Problem Statement v1.1.1," October, 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/001/01.01.01_60/gs_NFV-SEC001v010101p.pdf

[148] "Network Functions Virtualization (NFV) - NFV Security - Security and Trust Guidance v1.1.1," December, 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/003/01.01.01_60/gs_NFV-SEC003v010101p.pdf

[149] "Resolution 77 - Standardization work in ITU-T for software-defined networking," ITU-T World Telecommunication Standardization Assembly, Tech. Rep., November 2012. [Online]. Available: http://www.itu.int/en/iTU-T/wtsa12/Documents/resolutions/Resolution%2077.pdf

[150] ITU-T SG13 Future Networks - Questions Under Study. [Online]. Available: http://www.itu.int/en/ITU-T/studygroups/2013-2016/13/Pages/questions.aspx

[151] IETF FORCES (Forwarding and Control Element Separation). [Online]. Available: http://datatracker.ietf.org/wg/forces/

[152] IETF PCE (Path Computation Element). [Online]. Available: http://datatracker.ietf.org/wg/pce/

[153] IETF ALTO (Application-Layer Traffic Optimization). [Online]. Available: http://datatracker.ietf.org/wg/alto/

[154] IETF Netmod (NETCONF Data Modeling Language). [Online]. Available: http://datatracker.ietf.org/wg/netmod/

[155] IETF NetConf (Network Configuration). [Online]. Available: http://datatracker.ietf.org/wg/netconf/

[156] IETF NVO3 (Network Virtualization Overlays). [Online]. Available: http://datatracker.ietf.org/wg/nvo3/

[157] IETF I2RS (Interface to the Routing System). [Online]. Available: http://datatracker.ietf.org/wg/i2rs/

[158] IRTF SDN Research Group. [Online]. Available: http://irtf.org/sdnrg

[159] D. Meyer, "The Software-Defined-Networking Research Group," *IEEE Internet Computing*, pp. 84–87, 2013.

[160] "OpenFlowSec." [Online]. Available: http://www.openflowsec.org/SDNSuite.html

[161] "OpenDaylight AAA." [Online]. Available: https://wiki.opendaylight.org/view/AAA:Main

[162] "OpenDaylight Defense4All." [Online]. Available: https://wiki.opendaylight.org/view/Project_Proposals:Defense4All

[163] "OpenDaylight SNBI." [Online]. Available: https://wiki.opendaylight.org/view/SecureNetworkBootstrapping:Main

[164] "Network Functions Virtualization - Introductory White Paper," October, 2012. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[165] "OpenStack Cloud Software." [Online]. Available: www.openstack.org

[166] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[167] "ProtoGENI," 2014. [Online]. Available: http://protogeni.net/

[168] C. Hall, D. Yu, Z. li Zhang, J. Stout, A. Odlyzko, A. W. Moore, J. Camp, K. Benton, and R. Anderson, *Collaborating with the enemy*

on network management, ser. Security Protocols XXII. Springer, 2014, pp. 154–162.

[169] M. Moshref, A. Bhargava, A. Gupta, M. Yu, and R. Govindan, "Flow-level state transition as a new switch primitive for SDN," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 61–66.

Sandra Scott-Hayward (S'09-M'13) is a Senior Engineer in the Network Security research group at the Centre for Secure Information Technologies (CSIT), Queens University Belfast. She has experience in both research and industry, having worked as a Systems Engineer and Engineering Group Leader with Airbus before returning to complete her Ph.D. at Queens University Belfast. At CSIT, Sandra leads research and development of network security architectures and protocols for software-defined networks (SDN). Sandra is a Research Associate of the Open Networking Foundation (ONF) and actively participates in the ONF Security Working Group. She received an Outstanding Technical Contributor Award from the ONF in February 2015.

Sriram Natarajan is a Senior Research Engineer at Deutsche Telekom - Silicon Valley Innovation Center (T-Labs), USA. Dr. Natarajan is engaged in research in the areas of Software-defined Networks, Network Function Virtualization, and Network Security. Dr. Natarajan serves as the Vice-Chair of Security group at Open Networking Foundation. Prior to joining T-Labs, he worked as a Senior Researcher for NTT Innovation Institute Inc, where he designed and developed network services for NTT's SDN solutions. Sriram holds a MS and a PhD in Electrical and Computer Engineering from University of Massachusetts, Amherst. During his PhD, Sriram worked on addressing security issues in network virtualization.

Sakir Sezer (M) received the Dipl. Ing. degree in electrical and electronic engineering from RWTH Aachen University, Germany, and the Ph.D. degree in 1999 from Queens University Belfast, U.K. He is currently Research Director and Head of Network and Cyber Security Research at CSIT and holds the Chair for Secure Information Technologies at Queens University Belfast. His research is leading major (patented) advances in the field of high-performance content processing and is currently commercialized by Titan IC Systems. He has coauthored over 160 conference and journal papers in the areas of network security, content processing, malware detection, and System on Chip. Prof. Sezer has been awarded a number of prestigious awards including InvestNI, Enterprise Ireland and Intertrade Ireland innovation and enterprise awards, and the InvestNI Enterprise Fellowship. He is also cofounder and CTO of Titan IC Systems and a member of the IEEE International System-on-Chip Conference executive committee.