Journal of Network and Computer Applications 131 (2019) 89–108

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Review

The application of Software Defined Networking on securing computer networks: A survey

Rishikesh Sahay, Weizhi Meng *, Christian D. Jensen

Dept. of Applied Mathematics & Computer Science, Technical University of Denmark, Lyngby, Denmark

ARTICLE INFO

Keywords:
Software Defined Networking
Attack detection and mitigation
Network security
Middlebox management
Traffic management
Policy management
Traffic engineering
Smart grid security

ABSTRACT

Software Defined Networking (SDN) has emerged as a new networking paradigm for managing different kinds of networks ranging from enterprise to home network through software enabled control. The logically centralized control plane and programmability offers a great opportunity to improve network security, like implementing new mechanisms to detect and mitigate various threats, as well as enables deploying security as a service on the SDN controller. Due to the increasing and fast development of SDN, this paper provides an extensive survey on the application of SDN on enhancing the security of computer networks. In particular, we survey recent research studies that focus on applying SDN for network security including attack detection and mitigation, traffic monitoring and engineering, configuration and policy management, service chaining, and middlebox deployment, in addition to smart grid security. We further identify some challenges and promising future directions on SDN security, compatibility and scalability issues that should be addressed in this field.

1. Introduction

Computer networks generally consist of a large number of network devices such as switches, routers, and middleboxes (i.e. devices which process traffic other than forwarding). A large number of servers and hosts are interconnected through these network devices and middleboxes. These network devices and middleboxes are vendor specific and they have proprietary solutions. Network operators are responsible for configuring each device to handle network and security events. Configuring these devices manually with low-level device specific syntax is a tedious, complex, time consuming and error prone task, as network operators are required to be present all the time to configure these devices. This is a main reason for network downtime (Open Networking Foundation, 2008; Colville and Spafford, 2010), because automated configuration and modification are absent in the traditional IP networks (Benson et al., 2009a). However, considering the current network dynamics, it is very important that a network can adapt to the current status automatically. But, in the traditional IP network it is difficult to configure the network dynamically according to the current status.

To make it even more difficult, in the traditional network, control and data plane are integrated inside the network devices, which may reduce flexibility and dynamicity. Moreover, the lack of programmability and centralized control in the traditional network makes it difficult to deploy new services without halting the ongoing services (Chen et al., 2009). The scenario could be made even worse because of the enormous size of a network (Kim et al., 2011). The expanding network size and the heterogeneity can further increase the difficulty. These problems have demanded a new approach to support the deployment of network devices and applications.

The recent emergence of Software Defined Networking (SDN) addresses these issues by decoupling the network control from the data plane of the network devices (Open Networking Foundation, 2012, 2013). In SDN, network intelligence is logically centralized in a software based entity, called *controller*, and network devices like switches and routers behave as a simple forwarding device. Forwarding devices contain the flow rules to process the incoming packets based on match fields mentioned in the flow tables (such as source IP, destination IP, protocol, etc.). These forwarding devices can be programmed by the controller using a standard interface (such as OpenFlow (McKeown et al., 2008a), NETCONF (Enns et al., 2011), and ForCES (Halpern et al., 2010)). Moreover, network operators can define the high-level network policy at the controller, which can be enforced in the switches by the applications running on the controller.

* Corresponding author.
E-mail addresses: risa@dtu.dk (R. Sahay), weme@dtu.dk (W. Meng), cdje@dtu.dk (C.D. Jensen).

https://doi.org/10.1016/j.jnca.2019.01.019
Received 25 September 2018; Received in revised form 13 December 2018; Accepted 14 January 2019
Available online 21 January 2019
1084-8045/

In addition, as SDN controller can collect the flow statistics from the switches in the network, it can provide real-time global network status by analyzing flow statistics i.e., via the applications deployed at the controller (Tootoonchian et al., 2010; van Adrichem et al., 2014; Mann et al., 2012; Zhong et al., 2017). This can dramatically simplify the network architecture and provide effective network management.

Motivations. The global visibility and programmability of the SDN controller open up new avenues for network security. Recently, cybersecurity has received more attention from academia as well as from industry (CISCO, 2018; Mahimkar et al., 2007; F5, 2014; Bawany et al., 2017). According to the report of Symantec, cyber attacks continue to grow rapidly (Symantec, 2018). Obviously, deployment of services in the network without adequate security may cause a lot of risks and vulnerabilities, which can be exploited by attackers (Bisong and Rahman, 2011). Moreover, security is a major concern not only in enterprise and ISP, but also in different types of networks (Wu et al., 2018a; Aydeger et al., 2015). To tackle the challenges posed by the growing cyber attacks, there is a need to dynamically configure and deploy the security policies on-demand. Currently, most network operators rely on statically deployed devices to protect the network from cyber attacks (Arbor Networks, 2016; Fayaz et al., 2015). It is a time consuming process and causes network operators to either over provision or under provision the resources. In this paper, our interest is to investigate how SDN can help handle these concerns.

Regarding SDN, Open Networking Foundation (ONF) that is an industry driven organization has been created by many service providers and different network vendors to help promote software-defined standards (Open networking foundation). SDN now has gained significant attraction from both academia and industry. Most of the vendors like IBM and Hewlett-Packard have already launched switching devices which support OpenFlow protocol (Anon. 1; Anon. 2). Google has also deployed SDN to connect its data centers (Mandal, 2015). On the other hand, academic researchers are also actively involved in the development and deployment of SDN (Yap et al., 2009; Openflow network research center).

Comparison with similar surveys. A few recent papers have surveyed about protecting SDN networks like (Li et al., 2016; Alsmadi and Xu, 2015; Scott-Hayward et al., 2013; Ali et al., 2015; François et al., 2014; Kreutz et al., 2015). Surveys related to security of SDN and OpenFlow can be found in Li et al. (2016), Alsmadi and Xu (2015) and Scott-Hayward et al. (2013). In François et al. (2014), the authors review the work related to firewall deployment in the SDN network for security purpose. Kreutz et al. (2015) presented a comprehensive survey on SDN. They mainly discussed main concepts of SDN, how it differs from traditional networking. Moreover, it also presented a short discussion on the applications developed using SDN to improve the security. However, it lacks a clear categorization of the different types of applications developed to improve the security using SDN. Ali et al. (2015) go a step further by categorizing the SDN-based security research. However, their surveys mainly focus on threat detection, mitigation and network verification applications developed using SDN. Overall, existing surveys lack a thorough discussion on different categories of applications in relation to attack detection using machine learning and entropy based approach, traffic monitoring, vulnerability detection, traffic engineering, middlebox deployment, policy based security management, and service chaining though SDN. Moreover, there is a need to summarize releted studies on smart grid security through deploying SDN, as security of a smart grid infrastructure is a challenging task. As a result, there is a need to provide a more complete overview of ongoing research efforts and related activities.

In this paper, we aim to provide a more extensive survey on how to apply SDN for securing computer networks. In particular, we categorize the application of SDN for achieving network security into different categories. First, in Section 2, we present the architecture of SDN and its unique characteristics. In Section 3, we survey attack detection using machine learning and entropy based mechanism, vulnerability detection, attack mitigation, traffic monitoring, traffic engineering and dynamic configuration applications that are designed using SDN. Section 4 then presents security policy management, service chaining, middlebox deployment applications that are developed using SDN. In Section 5, we summarize the studies related to smart grid security via SDN. In Section 6, we discuss the main challenges and provide future directions regarding how to solve these challenges and enhance network security by using SDN. Finally, we conclude the work in Section 7.

2. Software-defined networking architecture

Computer networks are mainly comprised of three planes: forwarding, control and management plane. The forwarding plane consists of networking devices such as routers and switches which are responsible for forwarding traffic. The protocols used to deploy the rules in the forwarding tables of the data plane devices reside at the control plane. The management plane consists of the network and security policies to manage the network. Control plane enforces these policies in the forwarding plane. In the traditional computer networks, the control and data plane are tightly coupled with each other and integrated in the same networking devices. It makes the network rigid and static in nature.

Because of strong coupling between control and data plane in traditional computer networks, it becomes a very tedious and complex task to develop and deploy new network applications (Benson et al., 2009b). It requires modification in the control plane of all network devices through some hardware upgrades. Thus, new network and security features are often deployed in the network through the introduction of middleboxes such as Intrusion Detection Systems (IDS), firewalls, load balancer, etc. Generally, these middleboxes are placed statically at some key locations in the network, which makes it difficult to dynamically reconfigure them at run-time depending on the network conditions. Furthermore, to manage the network, administrators have to maintain specialized team to configure vendor specific solutions.

Software-Defined Networking (SDN) is a new networking paradigm which separates the control plane from the data plane. It is also comprised of the three planes: the data, control and management plane. To represent the work on SDN (McKeown et al., 2008b), the term was coined at Stanford University. Since then, it has attracted attention of researchers from both academia and industry. Fig. 1 illustrates the SDN architecture. The separation of control and data plane enables to deploy network applications based on the current network requirements. Many novel network and security applications based-on SDN have been proposed (Shin et al., 2013; Fayaz et al., 2015; Li et al., 2014; Krishnan and Durrani, 2014; Mehdi et al., 2011a). The main reason for researchers to have interests in SDN is mainly because it provides centralized control, where policies can be expressed for various network conditions and these policies can be enforced in the data plane through southbound API. Moreover, SDN offers global visibility and programmability which allows to configure the networking devices automatically based on the current network status.

As shown in Fig. 1, the SDN architecture consists of three distinct layers:

- The data plane layer consists of the network devices such as routers/switches, Intrusion Detection System (IDS), computers, firewall devices that perform packet forwarding and filtering.
- The Southbound interface provides an interface for communication between the data plane devices and the control plane elements. It specifies the communication protocol between data plane devices and the SDN controller.
- The control plane configures the network devices in the data plane through southbound interface which offers the communication between the data and control plane. It is also called as a net-

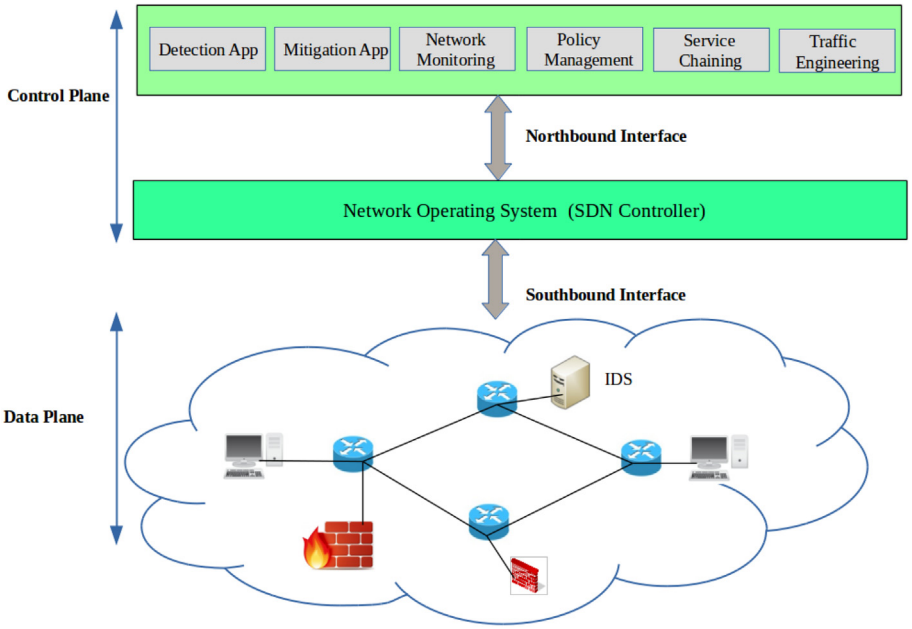

Fig. 1. SDN Architecture.

work operating system (NOS) or brain of the network. All the logic resides in the applications and the SDN controller, which together is called as the control plane.
- The northbound interface offers an interface to developers for designing applications. It hides the low-level implementation details to program data plane devices.

Therefore, an SDN architecture is characterized by the following key attributes:

- *Logically Centralized Controller and Network-Wide Visibility:* Data plane devices are connected to a centralized controller. The SDN controller can also send queries to data plane devices to get the flow statistics to infer the network status. With a centralized controller and the knowledge of global network status, decision making is facilitated, as opposed to legacy networks where nodes are unaware of the overall state of the network.
- *Programmability:* In SDN networks, data plane devices are controlled by the applications deployed at the controller. It offers an approach to introduce new network and security functions. Several programming languages have been proposed to enable this feature (Voellmy and Hudak, 2011; Foster et al., 2010; Voellmy et al., 2012).
- *Abstraction:* The applications deployed at the controller are offered an abstract view of the network. Applications can specify the desired network behavior through high-level policy languages (Voellmy and Hudak, 2011; Foster et al., 2010; Voellmy et al., 2012). These high-level configurations can be mapped into data plane configuration by the SDN controller. Moreover, the simplified data plane offers the flexibility to add new features, like DevoFlow (Curtis et al., 2011), NetFPGA (Naous et al., 2008; Al-Fares et al., 2010).
- *Flow-based Management:* Forwarding decisions in the SDN switches are taken on per flow basis. The basic characteristics of SDN is to forward the flow to the controller, when the network devices do not have the rules to handle these flows. This feature enables the network administrator to deploy the rules when it is required. Moreover, it enables to process the flow dynamically based on the various conditions in the network (Al-Fares et al., 2010).
- *Dynamicity:* SDN provides flexibility to accommodate changes for more dynamicity. Data plane devices can be reconfigured easily depending on the changing conditions in the network. It enables to deploy the network and security applications for on-demand services in the data centers and service provider network.

3. SDN enabled security mechanisms

There are numerous research studies on how to enhance the network security using SDN technology. In this section, we summarize research on the following aspects of SDN applications: dynamic configuration, attack detection, attack mitigation, traffic monitoring, and traffic engineering.

3.1. Dynamic configuration using SDN

The centralized control plane and the connection between the control and data plane through southbound interface is well suited to achieve the dynamic configuration in the network. The controller can send control messages to data plane devices. With a centralized controller and the knowledge of global network status, decision making is facilitated, as opposed to legacy networks, where nodes are unaware of the overall state of the network.

The FRESCO (Shin et al., 2013) framework proposed by Shin et al., offers a platform for rapid design of detection and mitigation modules. Detection and mitigation components are programmed and linked as modular libraries to provide a defense in the network. Upon detection of an attack in a network by detection modules, the FRESCO mitigation module generates flow rules to mitigate the attack. The framework consists of application layer (to build applications) and a security enforcement kernel which implements the action from the application. The FRESCO's application layer modules are developed in Python and run over the NOX OpenFlow controller. The Security Enforcement Kernel (SEK) is integrated with the NOX OpenFlow controller and offers features upon which FRESCO relies for the enforcement of the rules.

Application of the framework is demonstrated with two examples. In the first example, the authors have shown the working of honeynet to detect the malicious scanner by composing two modules together. Scan detection module detects a malicious scanner and then redirects all the malicious flow towards a honeypot for processing. Then, hon-

eypot responds to the scanner, which is unaware that all of its flows are processed by the honeypot. In the second example, the authors demonstrate that legacy security devices like BotHunter, DPI can be integrated with the FRESCO. These security devices monitor the network to identify malicious traffic. Upon detection of malicious traffic, alerts generated from these security applications are forwarded to the mitigation module, which generates the OpenFlow rules to mitigate the attacks.

F.Graur (2017) presented a mechanism to dynamically reconfigure the network in the Internet of Things. In this approach, an SDN controller is developed using python language to perform the network configuration. Controller dynamically computes the path when new flows arrive at the controller. The controller has two modules: NetModel and ModFloodlight. The NetModel contains the network details, and communicates with the Floodlight controller to install the rules for the new flows. The ModFloodlight takes the list of switches, source and destination IP address pair to deploy the rule for the flow.

Casado et al. (2007) proposed a centralized network architecture called as Ethane to configure the network. Network and security policies are expressed at the centralized controller. When the flows arrive in the network, rules are provided according to the host or service it wants to connect. Moreover, it introduced Ethane switches which consist of forwarding hardware to track flows. Ethane switches can be deployed along with the ethernet switches in the network.

3.2. Attack detection

Cyber attack detection is an important aspect that needs to be considered by network administrator. It is an important element for network security. With the help of effective attack detection mechanism, network administrators can handle different services and deploy network resources for mitigation in a more efficient way. The SDN paradigm provides a global visibility in the network which is ideally suited for attack detection applications. The SDN controller can periodically collect flow statistics for real time flow analysis from the data plane devices. It helps in analyzing the current network status for real time mitigation of attack traffic. In this section, we review both entropy based and machine learning based approaches for detecting attacks in SDN environments.

3.2.1. Entropy and threshold based detection in SDN

Giotis et al. (2014a), introduced an anomaly based detection mechanism using OpenFlow and sFlow statistics collection technique. The framework uses entropy based anomaly detection technique and makes a comparison between OpenFlow and sFlow flow mechanism over SDN. The authors implement the entropy based detection and the threshold random walk with credit based connection rate limiting (TRW-CB) algorithms. Experimental results show that the average number of flows required under the sFlow collection mechanism with entropy based method is 217 flows. However, with the OF based approach the required number of flows is 5184 flows. The same is the case with TRW-CB algorithm. Average number of flows required is 217 with sFlow mechanism; however, with the OF based approach the number of flows required is above 2000. Therefore, the authors illustrate that the sFlow traffic collection mechanism is better than the native OF (OpenFlow) traffic collection mechanism. The authors conclude that the lack of sampling in the OF flow statistics collection mechanism increases the number of flows required for analysis. However, it focused on layers (L2-L3) without looking into the packet contents.

In YuHunag et al. (2010) a two-stage rate based detection mechanism is used to detect the DDoS attacks. Network administrator sets a threshold, and once the traffic exceeds the threshold, then the second stage detection would be activated. For instance, the authors in their experimentation set 3000 packets for every 5 s as the first stage detection. Once the traffic exceeds this threshold, then the second stage threshold of 800 Packets Per Second (PPS) is activated. If the traffic continues at the 800 PPS for 5 s, then DDoS defense application starts filtering the traffic.

In Mehdi et al. (2011a, 2011b), multiple anomaly detection algorithms are implemented in order to validate their usability in small cases. The authors proposed that the anomaly detection algorithm can run at the border router of the home network. Four anomaly detection algorithms were implemented over the NOX SDN controller. These algorithms are Threshold Random Walk with Credit based Rate Limiting (TRW-CB), Rate Limiting, Maximum Entropy Detector, and NETAD. For the validation, authors used low network traffic rates ranging from 60 to 12, 000 packets per second. Experimental results indicate that the algorithms perform well in the small enterprise network environment. However, when being tested at the ISP level, the algorithms did not perform well. It shows that the detection is far more accurate and efficient in the home network in comparison to the ISP network, because of the high volume of traffic in the ISP network.

An SDN based anomaly detection and mitigation mechanism is introduced in Giotis et al. (2014b). It relies on bidirectional sketch count algorithm to detect the attacks. It identifies the destination IP addresses with high asymmetric communication pattern depending on a threshold value to detect the asymmetry in communication. It uses a threshold value to detect the asymmetry in communication. Once the controller identifies the malicious flows, it instructs the switch to drop the malicious flows. Legitimate traffic is still forwarded to victim as compared to Remote Triggered Black Hole (RTBH) in which victim becomes unreachable for benign traffic. The proposed mechanism provides a modular approach for data collection, anomaly detection and victim identification and attack mitigation. In this mechanism, detection and mitigation are performed at the ingress location of the network. Once the malicious flow is identified, it can be blocked at the entry point in the network, which saves the network resources from collateral damage. The framework relies on BGP protocol to embed the incidence report. The dependence on BGP provides some implications. First, BGP is very complex and any modifications will challenge the deployment. Second, the exchange of incidence report between domains after BGP needs time for update. Therefore, the latency will increase with the number of hops between the victim and source of attack. While the framework does not validate the authenticity of exchange report.

Wang et al. (2015) introduced a distributed algorithm for entropy based anomaly detection. The authors argue that previous studies rely mainly on flow statistics collected from the flow tables of the switches to perform anomaly detection in the controller. However, in the large network, such collection mechanism may overload the communication channel between the switches and the controller. To address this issue, they introduced a flow statistics process in the switch, and proposed a lightweight entropy based DDoS flooding attack detection in the edge switch of the network, thereby reducing the flow collection overhead on the controller. In the proposed mechanism a probability distribution based on destination IP address is used to calculate the entropy. In comparison to earlier studies (Mehdi et al., 2011a, 2011b), it reduces the communication overhead and performs the detection at small time scale.

In NetFuse (Wang et al., 2013), a proxy device is deployed between the switches and the controller, which monitors the network load, and instructs switches to reroute flows which are causing congestion to NetFuse devices. It performs multi-dimensional flow aggregation based on threshold to identify group flows with suspicious behavior. Moreover, it uses adaptive control to limit the impact of aggressive flows on legitimate traffic. However, attacks can still compromise configuration messages sent by the controller to the data plane by modifying their content.

StateSec (Boite et al., 2017) proposed an entropy based algorithm to detect threat such as DDoS and port scans. It relies on in-switch processing capabilities to detect DDoS attacks. Entropy evaluates the unpre-

Table 1
Entropy and Threshold based Detection Applications over SDN.

Detection Application	Detection Method	Drawbacks
Combining OpenFlow and sFlow (Giotis et al., 2014a)	Compares the OF flow statistics collection with sFlow based collection approach on entropy based method	Focus on layers (L2-L3), does not look into the packet contents
Two stage rate based detection (YuHunag et al., 2010)	Two stage threshold based detection is used	Only rely on packet rate threshold, set threshold is very low;
Traffic Anomaly Detection in SDN (Mehdi et al., 2011a, 2011b)	Threshold Random Walk, Rate limiting, Maximum entropy detector and NETAD algorithms were tested	Does not perform the detection at small time scale
Remote Triggered Black Hole (RTBH) (Giotis et al., 2014b)	Uses bidirectional count sketch algorithm and asymmetric communication pattern to detect the attacks based on threshold	Reliance on BGP protocol impacts the exchange of incidence report between domains
NetFuse (Wang et al., 2013)	Uses multi dimensional flow aggregation to detect and safeguard the network from traffic overloading	Configuration of messages between the controller and data plane is not considered
Radware (McGillicuddy, 2013)	Makes a baseline profile of network traffic and then monitors the network for anomalies	Does mention how security alerts are exchanged
StateSec (Boite et al., 2017)	Uses entropy based algorithm for detecting DDoS attacks and port scanning	Does not provide any implementation of port scanning attack

dictability of distribution. Abrupt variations in the measured entropy enable detecting anomalies based on the traffic features. The authors claim that this mechanism can be used to detect port scanning attack; however, they do not provide any implementation of that.

Industrial solutions have also been proposed to detect the attacks. Radware (McGillicuddy, 2013) has developed DefenseFlow a solution to detect the Distributed Denial of Service (DDoS) attacks. The application maintains a baseline profile of the network traffic and then monitors the real-time traffic for DDoS attacks. It relies on behavioral parameters such as packet rate, connection rate, bandwidth, connection distribution and average packet size, etc. Upon detecting an attack, it notifies the controller to reroute the attack traffic to specialized devices deployed in the network to process the traffic.

Table 1 shows the entropy and threshold based DDoS detection applications deployed in SDN, including their methodology and drawbacks. The detection method describes the mechanism used to detect the attack. Drawbacks specifies the limitation of the mechanism.

3.2.2. Machine learning based detection

Peng et al. (2018) proposed a method for detecting anomalous flow relying on SDN technology. In this mechanism, the flow collection module of the SDN controller gathers the flow table information from the switches and extracts different features of the flow. Then, anomaly detection mechanism processes the flow features and performs the flow classification by means of the DPTCM-KNN algorithm. However, in the proposed mechanism, the controller collects flow statistics from the switches in every 10 s, which can cause processing overhead on the controller.

In Meng et al. (2018), the authors proposed a trust-based approach based on Bayesian inference to identify malicious devices in a healthcare environment, by leveraging both packets' status and device profile.

The lightweight DDoS detection using OpenFlow proposes to use Self Organizing Maps (SOM) to detect attacks (Braga et al., 2010). The authors used SOM, which is an unsupervised neural network method, to classify the network traffic as normal or abnormal. In this mechanism, an SDN controller periodically collects the traffic statistics from the switches. It extracts the features (e.g, average of packet per flow, average of bytes per flow, average duration per flow) from flow entries of all switches, and then the extracted features are forwarded to the classifier module for further processing. The classifier module analyzes whether traffic is legitimate traffic or DDoS flooding traffic. However, the framework depends on the flow statistics from all the OpenFlow switches in the network, which may cause some processing overhead at the controller.

Xie et al. (2018) surveyed how machine learning algorithms are applied in SDN, from the perspective of security, resource optimization, traffic classification and Quality of Service (QoS) prediction. The authors argue that the machine learning techniques can bring intelligence to the SDN controller as it enables the SDN controller to autonomously learn to make optimal decisions to changing network environments. Different types of machine learning algorithms and their applications are presented in the paper.

In Ashraf and Latif (2014) the authors analyzed different machine learning techniques which can be used to tackle the problems of intrusion and DDoS attacks to SDN. The paper provides the pros and cons of surveyed mechanisms in tackling the intrusion detection problems in an SDN environment.

Glick and Rastegarfar (2017) studied the traffic flow scheduling problem in a hybrid data center network. Machine learning methods are used to perform elephant flow traffic classification at the boundary of the network. Classification at the edge of the network reduces the burden on the SDN controller and achieves the scheduling with minimum number of configurations. However, it does not provide network wide large scale data extraction and management across distributed data plane, with many switches and controller.

EUNOIA (Song et al., 2017) is a threat aware system to perform detection and make response to attacks in SDN. The framework is composed of data preprocessing, predictive data modeling, decision making and response system. To select appropriate feature sets, a forward-feature selection mechanism is used by the data preprocessing module. Then, random forest and decision tree algorithms are used by predictive data modeling module to detect suspicious and malicious activities. Depending on the alerts, the decision making and response module compute the routing path and install the flow rules for different flow types. The experimental results show that with the forward feature selection approach, the proposed framework is able to reduce the data preprocessing time and maintain high detection accuracy. However, it reports that when seven hosts send 1 GB of traffic to a target, it takes several thousand seconds to process and select the routing path for the traffic. That is, scalability is a major concern for this framework.

ATLANTIC (da Silva et al., 2016) is a management framework to perform the task of anomaly detection, classification and mitigation in SDN environment. It performs the anomaly detection and classification in two phases: (1) a lightweight phase, in which low computation mechanisms are applied to spot suspicious and malicious flows, and (2) a heavyweigth phase, where the support vector machine is used to classify the abnormal flows. However, for this method, the controller suffers the overhead of polling all the switches for the statistics.

In Niyaz et al. (2017), the authors proposed a deep learning based DDoS detection system for SDN environment. Stack autoencoder based

Table 2
Machine Learning based Detection Applications over SDN.

Detection Application	Detection Method	Drawbacks
DPTCM-KNN (Peng et al., 2018)	Flow classification is performed using DPTCM-KNN algorithm	Collects the flow statistics every 10 s from all the switches which can overload the controller
Lightweight DDoS Detection (Braga et al., 2010)	Self-organizing map with traffic features is used to detect the attack	Traffic statistics is collected from all the switches, overload the controller
Survey on ML Technique in SDN (Xie et al., 2018; Ashraf and Latif, 2014)	analyzes different machine learning techniques applied for intrusion and DDoS attacks to SDN	–
Flow Scheduling (Glick and Rastegarfar, 2017)	Neural network method is used to perform flow classification for scheduling	Detailed algorithm and implementation is not provided
EUNOIA (Song et al., 2017)	Is a threat aware system for detection and mitigation	Lack scalability with large number of hosts and traffic
ATLANTIC (da Silva et al., 2016)	management framework to perform the task of anomaly detection, classification	Processing overhead on the controller because of polling all the switches
Deep Neural Network Approach (Niyaz et al., 2017; Tang et al., 2016)	Neural network algorithms are used to perform detection and classification of flows	Processing overhead on the controller because of polling all the switches
ATHENA (Lee et al., 2017a)	Offers API for implementing different anomaly detection mechanisms	Lack adaptive measurement for resource optimization

deep learning approach and softmax classifier for unsupervised learning and classification has been developed, respectively. The experimental results show that the 5-class classification of the NSL-KDD dataset produced an f-score of 75.76 percent and accuracy of 69 percent. However, as every packet has to be collected for extracting features, this approach may limit the performance of the controller in large networks.

Tang et al. (2016) presented a deep neural network approach for conducting intrusion detection in an SDN environment. The authors designed a deep neural network with an input layer, three hidden layers and an output layer. In the proposed system, the controller collects the flow statistics from all the switches in the network and forwards it to the detection module for analysis and detection. The experimental results report an accuracy of 75.75 percent using the NSL-KDD dataset.

Lee et al. (2017a) proposed a framework called Athena for scalable anomaly detection using SDN. The authors argue that prior studies mostly deploy the detection mechanism at the egress boundaries in the network, but did not consider the network wide deployment with many switches and instances of controller. In addition, they mainly focused on applications designed for specific suspicious and malicious activities. To mitigate this challenge, Athena provides an API for implementing a wide range of anomaly detection applications across physically distributed control and data plane. It abstracts a complex data abstraction mechanism, reducing the programming effort needed to implement anomaly detection mechanism. Moreover, it offers a wide range of network features and detection mechanisms for deployment in large-scale SDN networks. Furthermore, it avoids the need for specialized hardware for the deployment of detection mechanisms. However, Athena does not provide adaptive measurement for resource optimization.

Table 2 shows the machine learning based DDoS detection applications deployed in SDN, including their methodology and drawbacks. Detection methodology describes the approach used to detect the attack. Drawbacks specifies the shortcomings or limitations in the mechanism. As shown in Table 2, most mechanisms may cause processing overhead on the controller and lack scalability in terms of network wide deployment. ATHENA (Lee et al., 2017a) tries to overcome this limitation but it lacks adaptive measurement for resource optimization.

3.3. Vulnerability detection

A thorough vulnerability assessment of SDN components is required because of increasing interest in the deployment of SDN. It helps understand the various attack surfaces in various components of SDN. In this regard, STS (SDN Troubleshooting System) (Scott et al., 2014) introduced a fuzzing technique which randomly injects events like link failures or packets into a network and finds seven new bugs in five different types of SDN controllers. The main focus of STS is identifying the minimal causal sequence associated with a bug.

Lee et al. (2017b) presented a framework called DELTA for vulnerability assessment in SDN environment. The framework offers an automated vulnerability assessment in SDN for diverse attack scenarios. The authors argue that prior studies provide limited coverage of the SDN attack surface, as they generally depend on specific SDN elements or network environments. To overcome this problem, the authors used a blackbox fuzzing technique to detect unknown vulnerabilities in an SDN environment. Blackbox fuzzing technique randomizes message-input values to detect vulnerabilities. However, such random fuzzing neither considers the message structure nor protocol semantics. Moreover, it mainly focuses on SDN controller's northbound interface without considering the vulnerabilities in the southbound interface.

OFTest (Floodlight Project, 2016) and FLORENCE (Github - snrism/florence-dev, 2016) test OpenFlow switches using manually written tests to check the conformity to openflow specification. In particular, FLORENCE tests 18 different scenarios and OFTest covers a few hundred. However, these tools do not consider the controller, that is, they are unable to find bugs and attacks in the controller. Furthermore, test cases are designed manually, hence it is difficult to evaluate the coverage of test cases systematically.

Yao et al. (2014) introduced a formal model and blackbox testing approach for SDN data plane. The authors proposed a model of pipelined extended finite state machine (Pi-EFSM) to define multiple-level pipeline of SDN data plane. The hierarchical test-generation strategy could reduce state space explosion to some extent. However, this tool does not consider the vulnerabilities in the controller.

BEADS (Jero et al., 2017) is a framework which automatically generates test cases and find attacks in an SDN environment. The framework decouples the attack strategy generation from the testing of a strategy in the SDN environment. It automatically generates and tests thousands of cases comprising of malicious switches which do not comply with the OpenFlow protocol and malicious hosts which do not obey the ARP protocol. The framework combines the technique such as byzantine fault injection, semantically aware test case generation and black box testing to evaluate the SDN environment. The framework does not need the code of switches or controller as required by most existing testing tools on SDN. Moreover, it can test controller algorithms like routing, topology detection, and flow rule management.

Table 3
Vulnerability detection application in SDN.

Vulnerability Detection Application	Methodology	Drawbacks
STS (Scott et al., 2014)	Randomly injects events and packets into network to find bugs	Mainly focuses on testing paths of SDN data plane
DELTA (Lee et al., 2017b)	Uses a blackbox fuzzing technique to detect unknown vulnerabilities in SDN	Mainly focuses on controller's northbound interface, does not consider the vulnerabilities in southbound interface
OFTest (Floodlight Project, 2016)	Manually written blackbox testing approach is used	Does not consider the controller in the test cases
FLORENCE (Github - snrism/florence-dev, 2016)	Manually design the test cases to check the conformity to openflow specification	Does not consider the controller in the test cases
Pi-EFSM (Wang et al., 2013)	Uses formal model to define multiple-level pipeline of SDN data plane	Does not include controller in the scenario
BEADS (Jero et al., 2017)	Uses byzantine fault generation and blackbox testing approach to generate the test cases	The method is mainly designed to detect bugs in the control plane

Table 3 shows the vulnerability detection mechanisms based on SDN, including their methodology and drawbacks. We found that most mechanisms could perform the vulnerability detection either in the control plane or in the data plane.

3.4. Attack mitigation

The programmability and logically centralized controller of the SDN is well suited for mitigating attacks in the network. Flow rules in the switches can be modified on fly for mitigation purpose. Moreover, security alerts can be shared between different domains to mitigate the attacks in a collaborative manner. Recently, there have been some initiatives to design mechanisms for effectively mitigating attacks using SDN.

In Drawbridge (Li et al., 2014), the authors introduced a framework for mitigating attack traffic in the ISP network. The authors argue that the customers can subscribe to traffic engineering services provided by ISPs. It relies on the assumption that the customer's controller can share the mitigation rules with the ISP's controller for the deployment. The main aim of the framework is to avoid the unwanted dropping of customers' traffic by their ISP because of congestion. In the framework, detection is performed by the end-hosts to share the rules with the ISP controller for mitigation. Upon receiving the rules, the ISP controller checks the validity of the rules before deploying them in its switches. The framework enables the customers to be adaptive in dealing with the dynamic traffic.

The system introduced by SENSS (Yu et al., 2014) offers an interface to request for attack mitigation. Upon detection of attack by the victim network, it sends a message to the ISP with the details of the traffic and its route. The victim can ask the ISP to filter or modify routes of the attack traffic coming to its network. The important features of SENSS are summarized as follows:

- The SENSS offers a system to allow the victim to directly request for the security services such as mitigation from remote ISPs. Victim requests the ISP for mitigating the traffic which belongs to the victim's address space to avoid the conflict with different customers of the ISP;
- In the framework, the ISP provides an interface for victims to request for the mitigation services such as traffic filtering, rerouting from ISPs, etc,
- Moreover, victims can request multiple ISPs to traceback the source of the attack and perform the mitigation. However, this framework requires multiple ISPs to collaborate with each others.

The Bohatei (Fayaz et al., 2015) architecture leverages SDN and network function virtualization (NFV) capability for DDoS defense. In this architecture, the authors leveraged the NFV approach to instantiate the defense VM (Virtual Machine) at the required location in the network. The architecture can dynamically steer traffic towards instantiated VMs in the network. It offers an ISP centric deployment, where an ISP provides DDoS-defense-as-a-service to its customers. In this architecture, the ISP deploys some anomaly detection systems to detect attacks. Upon detecting attacks, an estimation is performed for the suspicious traffic. The resource manager module in the architecture uses the estimates to decide the type, number and location of VMs to be instantiated. It optimizes the network resources using two algorithms: (1) data center selection algorithm; (2) and server selection algorithm in the data center. The data center selection is a greedy algorithm that selects the data center for routing of suspicious traffic. The server selection algorithm tries to assign the servers with higher capacities to process the suspicious traffic in the selected data center. Then, forwarding rules are deployed in the switches to forward the traffic. The experimental results showed that the system is able to handle attacks of hundreds of Gbps. It enables to mitigate the DDoS traffic in less than 1 min. However, this framework can cause processing overhead on the ISP side, since it requires the ISP to perform the detection process on behalf of their customers. In practice, ISPs often have hundreds and thousands of customers to manage.

Sahay et al. (2015), proposed a framework for collaborative DDoS mitigation. The proposed framework allows the customers to request DDoS mitigation service from their ISP. Upon request of the customers, the ISP can redirect the suspicious and malicious flows towards the middleboxes for further processing. However, the framework assumes that the SDN controller of the customer and the ISP can securely communicate with each other. In addition, the authors did not provide any implementation of the framework.

The ArOMA (Sahay et al., 2017a) provides a framework for mitigating DDoS attacks in an automated way. It leverages the centralized manageability and programmability features of SDN to automatically mitigate DDoS attacks in the ISP network. The framework bridges the gap between different security functions ranging from monitoring to detection and to mitigation. The framework enables the collaboration between ISPs and their customers on DDoS mitigation. The authors argue that the attack targets on the customer as well as the ISP by traversing the traffic through the ISP network. To mitigate this issue, the proposed framework offers a security API at the ISP controller, so that the customer can share the security alerts with the ISP for mitigation. Moreover, the framework provides a controller-to-controller communication for mitigating the attacks in a collaborative way. Then, the authors present an implementation and evaluation of the framework.

In Wichtlhuber et al. (2015), the authors presented an architecture based on SDN for collaboration between the Internet Service Providers (ISPs) and content distribution network (CDN) to manage high volume flows such as video traffic. In the architecture, the ISP deploys an appli-

Table 4
DDoS mitigation applications over SDN.

Attack Mitigation applications	Mitigation Method	Mitigation Location	Shortcomings
Drawbridge (Li et al., 2014)	sharing of mitigation rules by customer with their ISPs	ISP of customer or upstream ISP	No Implementation
SENSS (Yu et al., 2014)	Provides an interface on the SDN controller which enables the victim to request for mitigation and rerouting service from the ISPs	Upstream ISP of victim network	No details given how alerts are shared
Bohatei (Fayaz et al., 2015)	Instantiate the VMs in the network to steer the suspicious traffic for processing	ISP network	Requires ISP to perform detection on behalf of the customers, causes processing overhead
Sahay et al. (2015)	Collaborative framework for DDoS mitigation between the ISP and the customer	ISP network	Does not provide any evaluation
ArOMA (Sahay et al., 2017a)	Customer shares the alert with the ISP for mitigating attack in the ISP network	ISP network	considers a scenario of single customer and its ISP
CDN-ISP Collaboration (Sahay et al., 2017a)	ISP offers an application for information with CDN to manage the long flows	ISP network	Performance of the algorithm deciding on the migration of connections were not evaluated
DDoS Blocking Application (Lim et al., 2014)	It provides redirected address to server which is used to redirect the legitimate connection upon detection of attacks	Protect the server from botnets by redirecting traffic from ingress switch	Bots do not perform IP spoofing
Automated Malware Quarantine (AMQ) (Open Networking Foundation, 2012)	Upon malware detection isolates the infected devices in the network	It is deployed at the edge of the network	No detail given

cation at its controller, which can be used by the CDN, to access to the hidden information such as topology and load information in the ISP's network for optimizing and selecting QoS for the end-users. The mechanism allows the ISP to manage the rising amount of traffic emanating from CDNs to reduce the congestion in its network.

The DBA (Lim et al., 2014) architecture presented an SDN-based DDoS blocking technique to mitigate botnet based attacks. In this mechanism, the authors aim to protect a server from botnets. The server under protection builds a secure communication channel with the DDoS Blocking Application (DBA) deployed at the controller. Upon detecting an attack, server notifies the DBA about the attack. Then, the DBA provides a redirected address to server. The DBA maintains a pool of public IP addresses which can be used for redirection. Upon receiving the notification from the server, the DBA application provides a new address to the server. Then, the server notifies the legitimate clients to use the new address to access the service. It is not mandatory that the new address should be physically replicated, but can be in the same subnet. Redirected address information provided to the clients requires clients to perform some computation to protect these services from bots. In the architecture, the authors used CAPTCHA (Ahn et al., 2003) to protect the services running at the server. However, in their evaluation, bots are poorly programmed in this mechanism as it may not be the case in the practical scenario. Moreover, they assumed that in this architecture, the bots cannot perform an IP spoofing.

Open Networking Foundation (ONF) (Open Networking Foundation, 2012) has discussed the advantages of SDN in providing security in the data center. It leverages centralized intelligence and global network visibility of SDN for mitigation. It provides a use case where an Automated Malware Quarantine (AMQ) system detects and isolates the infected and insecure network devices before it can affect the network. Upon detecting an attack, it dynamically configures the policies for mitigation. It allows the network devices to join the network after attack is mitigated. However, in the traditional network, AMQ is deployed in the vendor specific hardware devices, which requires an expert to configure.

3.5. Some insights and lessons learned

The above related studies reveal that the SDN technology can help ease the mitigation of cyber attacks. It enables the collaboration between different network domains for mitigating attack traffic by deployig security API at the SDN controller. Different domains can collaborate with each other for mitigating attacks without revealing their topology. Dynamic configuration provided by the SDN, offers the mitigation application to configure the rules to either block or redirect the traffic towards the middleboxes for processing. Moreover, automation achieved by the DDoS mitigation application using SDN assists can provide a quick recovery from attacks. However, SDN controller also opens several new avenues for attackers. To mitigate this issue, security alerts should be shared between different domains in a secure way. Moreover, there should be good authentication schemes in place to authenticate the users who are sharing the alerts. Conflicts among rules should be resolved before deploying the rules for mitigation. Table 4 shows the DDoS mitigation applications, relevant mitigation method, mitigation location and the corresponding drawbacks. As mitigation methods have already been discussed earlier. Here we mainly describe the mitigation location and the shortcomings of these applications. (Li et al., 2014), Bohatei (Fayaz et al., 2015) and SENSS (Yu et al., 2014), Sahay (Sahay et al., 2015), ArOMA (Sahay et al., 2017a),CDN-ISP (Wichtlhuber et al., 2015) were deployed in the ISP network of the customers. Drawbridge (Li et al., 2014), SENSS (Yu et al., 2014) and Sahay (Sahay et al., 2015) did not provide implementation details and how alerts are shared. Bohatei (Fayaz et al., 2015) requires ISP to perform detection on behalf of its customers which can cause processing overhead at the SDN controller of the ISP. In ArOMA (Sahay et al., 2017a), mitigation was performed at the ingress location of the ISP network. However, this method only considers a scenario of single ISP and single customer network. Automated Malware Quarantine (AMQ) (Open Networking Foundation, 2012), and DDoS Blocking Application (Lim et al., 2014) provided the mitigation at the ingress point in the network without collaborating with upstream networks. However, in DDoS Blocking Application (Lim et al., 2014), bots are poorly programmed.

3.6. Traffic monitoring in SDN environment

Generally, in the traditional network, the traffic monitoring applications are deployed on a separate hardware or require special software configuration making it tedious and complex to manage. Also, precise measurement of traffic matrix in the traditional network is difficult because of the high number of data plane devices. For this challenge, it is found that the traffic matrix estimation problem can be minimized using SDN. OpenTM (Tootoonchian et al., 2010) offers

Table 5
Traffic monitoring applications over SDN.

Traffic Monitoring Applications	Methodology	Deployment location	Shortcomings
OpenTM (Tootoonchian et al., 2010)	provides the traffic estimation for source and destination pair in a network	Switches closer to destination is used for querying the flow statistics	Generates traffic matrix for offline use
OpenNetMon (van Adrichem et al., 2014)	It provides monitoring to determine whether end-to-end QoS parameters are achieved and forward the input to TE to compute the paths	Edge switches in the network are used to collect the statistics	Per flow monitoring and forwarding rules raise scalability issue
OpenSAFE (Ballard et al., 2010)	Forwards the first packet of flow to the monitoring application to get the information about the flow	Middleboxes are deployed in the path to monitor the traffic	Requires expensive hardware to perform monitoring
OpenSketch (Yu et al., 2013)	Provides 3-stage pipeline (hashing, filtering, counting) and measurement library to configure resources for varied measurement tasks	No details given	New protocol in OpenSketch requires upgrade in network devices
PaFloMon (Argyropoulos et al., 2012)	Provides a slice based monitoring in the SDN network	No details given	It only offers passive monitoring

a traffic estimation mechanism by leveraging the global visibility of SDN and flow based operations of the SDN switches. OpenTM provides a traffic estimation for origin-destination (OD) pairs in the network. Moreover, it relies on the routing information of the SDN controller to intelligently select the switches for querying flow statistics, therefore reducing the load on the switches. It presents five different strategies to query switches for flow statistics: a) polling the last switch, b) querying switches in the path of flow uniformly at random, c) querying switches closer to the destination with a higher probability, d) round-robin querying, and e) querying the least loaded switch. The authors found that the querying the last switch close to the destination could provide most accurate traffic matrix, but it may impose considerable workload on the edge switches.

In van Adrichem et al. (2014), Adrichem et al. presented an SDN based application to monitor per flow metrics such as throughput, delay, and packet loss in SDN networks. OpenNetMon monitors the end-to-end QoS parameters between pre-defined link to compute appropriate paths in the network. It polls edge switches at an adaptive rate that reduces the network and switch CPU overhead, and minimizes the number of queries between switches and the controller.

The OpenSAFE (Ballard et al., 2010) architecture introduced a mechanism to route the traffic to monitoring applications. It leverages the fact that every first packet of a flow needs to be forwarded to the controller. Then SDN controller forwards the new flows to the traffic monitoring applications with an IDS. Moreover, it presents a language to enable the network administrators to easily manage and update the monitoring infrastructure. However, OpenSAFE requires expensive hardware to perform the monitoring.

OpenSketch (Yu et al., 2013) is an SDN-based monitoring mechanism, which separates the measurement data plane from the control plane. It offers a three stage pipeline (hashing, filtering, counting) which can be implemented with the switches to support different measurement tasks. The control plane provides the measurement library which automatically configures the pipeline and assigned resources to the varied measurement task. However, OpenSketch requires to update relevant network devices, making ISP reluctant to deploy this mechanism.

The Passive Flow Monitoring (PaFloMon) (Argyropoulos et al., 2012) aims at providing the slice based monitoring in the SDN networks. In this mechanism, network is segmented into different slices. It is motivated by the OFELIA (Openflow in Europe), which is an OpenFlow testbed in the Europe. The PaFloMon aims to enrich the OFELIA framework with slice aware monitoring.

Traffic monitoring applications are summarized in Table 5 in addition to a summary of methodology, deployment location and drawbacks of these applications.

3.7. Traffic engineering (TE) using SDN

Traffic engineering is very important as it helps in reducing the impact of congestion caused by attack traffic or overflooding traffic. In traditional networks, it requires to manually modify the rules or pre-deploy the rules in network devices for performing traffic engineering. Generally, if the link fails or topology changes, then it takes much time to converge. It is a very complex task for network operators. Furthermore, it makes the traffic engineering static in nature. In summary, the tight-coupling between the control and data planes, in addition to distributed control, makes the traffic management a cumbersome task in the traditional networks. However, the decoupled control and data plane in SDN can enable network administrators to express traffic engineering policies at the centralized controller. Moreover, network operator can leverage the global visibility of SDN to monitor the network status in the real time with the purpose of traffic engineering.

Shu et al. (2016) introduced a framework for traffic engineering using SDN. The framework comprised of two parts: (a) traffic measurement and (b) traffic management. Traffic measurement mainly includes monitoring and measuring the network status in real time for managing and controlling the traffic. The network measurement parameters include end-to-end network latency, topology connection status, bandwidth utilization, throughput, packet loss rates, etc. Real-time network status is depending on the QoS parameters. Based on the traffic measurement information and different QoS parameters, network traffic can be scheduled to meet the user's requirements in practice.

Cao et al. (2014) proposed a mechanism for offline traffic planing and online traffic routing in SDN networks. The SDN controller in the mechanism performs the policy lookup, flow steering, and route installation. The policy table at the controller determines the logical sequence of middleboxes for the traffic. The controller maps the logical sequence of middleboxes to the physical topology. Once the logical to physical path is mapped, then the SDN controller deploys the rules in the flow table of switches. The authors also proposed an algorithm for optimizing the resources such as the middlebox capacity to handle the estimated traffic. Furthermore, an online traffic steering mechanism is introduced to steer the flows upon their arrival depending on the policy requirements. For instance, a storage server in an enterprise network may require incoming traffic from the outside network to traverse the firewall, intrusion detection system (IDS) and a network address translator (NAT). Additionally, the amount of bandwidth needed is also considered for online traffic steering.

Agarwal et al. (2013) proposed a traffic engineering mechanism based on SDN by leveraging its characteristics such as logically centralized controller and global network view. The aim of this mechanism

is to dynamically manage the network traffic according to the requirements specified by the network administrators. The authors developed a Fully Polynomial Time Approximation Scheme (FPTAS) to compute the route for traffic steering depending on the optimization criteria to improve the bandwidth utilization, as well as to reduce packet losses and latencies. It considers the partial deployment of SDN switches in the network to route the traffic.

4. SDN based network and security services

In this section, we investigate the SDN-based network and security services including middlebox deployment, service chaining of different middleboxes and security policy management in an SDN environment.

4.1. SDN based middlebox deployment

Middleboxes (e.g., firewalls, NATs, DPI, WAN optimizer, IDS) are deployed in the network for critical services such as security and load balancing. But, in the traditional network settings, it requires careful design of network topology and intensive manual work from network operators to set up the rules in the middleboxes and in the switches to steer the traffic through a sequence of middleboxes. However, the advent of SDN technology provides a promising alternative for steering traffic through a sequence of middleboxes from a centralized control point. In this regard, SIMPLE (Qazi et al., 2013) introduced a system that allows network administrators to specify a policy to route the traffic through a logical sequence of middleboxes. It can be automatically translated into forwarding rules by considering both the physical topology and the middlebox resource constraints. The SIMPLE architecture has three key components:

1. The resource manager module takes as input a network traffic matrix, the topology of the network and policy requirements, to provide a set of middleboxes for processing the traffic.
2. The dynamic handler component correlates the incoming and outgoing traffic from the middleboxes.
3. The rule generation component takes the output from the resource manager and dynamic handler modules, and generates the rules to steer the traffic through a sequence of middleboxes.

Dynamic traffic modification performed by middleboxes such as NAT (Network Address Translator), IDS (Intrusion Detection System), and DPI (Deep Packet Inspection) causes the policy conflict between the controller and the data plane. Therefore, it becomes difficult to integrate the middleboxes in the SDN network. For this issue, FlowTags (Fayazbakhsh et al., 2014) architecture dynamically inserts a unique tag in the flow before it is processed by the middleboxes. This dynamic tag insertion in the flow by the middleboxes ensures: (i) the binding between the packet and its source, (ii) and it makes sure that the packet follows the path expressed by the central policy.

In Slick (Anwer et al., 2013), the authors pointed out that, in the traditional network, security administrators need to plan in advance the deployment of the middleboxes at some key locations in the topology. Because of the static and distributed management, it is difficult for the network operators to dynamically reconfigure the middleboxes based on demands. Thus, the network operator needs to over provision middlebox resources. To handle this problem, the authors proposed a *Slick* middlebox architecture. The main objective is to provide a mechanism by which network administrator can easily implement and deploy policies in the network. In the architecture, single policy is split into multiple executables that can run on multiple middleboxes. The *Slick* framework achieved this objectives by three main modules: (i) a protocol for coordination between the controller and the middleboxes, (ii) Single policy is split into multiple executables for deployment on multiple middleboxes, and (iii) an optimization algorithm at the controller dynamically deploys the middlebox functions and route the traffic.

4.2. Service chaining

Service chaining is performed to forward the traffic through a set of middleboxes in the network. It enables the network administrators to define how to process traffic through a sequence of middleboxes before reaching the destination. However, in the traditional network, it is difficult to steer the traffic dynamically for processing through these middleboxes due to the static configuration.

In StEERING (Zhang et al., 2013), the authors highlighted that managing middleboxes (also known as inline service chaining), such as NAT and firewalls, in data center or enterprise network is still a daunting task. Because the network operators still rely on the manual configurations at the low level to manage these devices, the whole network is insecure, i.e., can be bypassed by forwarding all the traffic through middleboxes or installing extra tunnels in the network. For this issue, StEERING allows to steer traffic using policies that are defined at the SDN controller. The StEERING framework consists of the data plane and control plane modules. The border switches classify the incoming traffic and forward it towards the next service in the service chain. Then the core switches in the network process the traffic depending on the L2 layer. The control plane of the framework consists of two modules. The first module is the SDN controller, which manages the switches in the network and deploys the flow rules in the OF switches. The second module is an optimization algorithm running at the controller, which periodically determines the best locations for the services to be deployed.

J.Blendin et al. (2014) proposed an SDN based service chaining architecture. The authors argued that most existing pre-configured and static service chains relied on fixed hardware and software configuration. While changing existing service chains requires intensive manual work that is time consuming for on-demand services. They pointed out that dynamic service chaining based on the SDN technology can improve the situation in two ways. First, service chains can be easily created from existing service functions. Second, service functions can be dynamically created in run-time. They then proposed a service chaining architecture based-on SDN, which consists of three layers: data plane, control plane, and application plane. The data plane layer contains the actual OpenFlow switches. The control layer contains the Service Function Chaining Router (SFCR) which is responsible to deploy the flow rules. The Service Function Chaining Controller (SFCC) resides at the application layer and is responsible for controlling all the components and service instance allocation, and offering high-level API. When a new flow arrives, it is provided a default service instance by the SFCC. The SFCC offers high-level API, which enables the network operators to define, modify and remove the service chains dynamically. The SFCR is instructed by the SFCC to update the service chain routing by updating the OpenFlow flow rules at the ingress and egress location of the service chain.

A dynamic service chaining architecture based on SDN is proposed in Scheid et al. (2016). It enables network administrators to express policies that govern the chaining of virtualized network functions (VNFs). Depending on the set of available VNFs, the framework creates a graph that represents the service chaining (SC). Moreover, this architecture introduces a high-level language to define the policies; thus, network operators do not need to be familiar with the low-level programming language to express the policies.

SECaaS framework (Migault et al., 2018) offers security services collaboration among different administrative domains. It leverages the SDN and service function chaining to improve the collaboration among different security service functions (SSF) to mitigate attacks. The framework enables the SSFs from multiple domains to negotiate and dynamically control the amount of resources allotted for the collaboration. This collaborative framework provides the distributed mitigation for handling large scale cyber attacks.

Table 6
Service chaining for security services.

Service Chaining	Short description	Shortcomings
StEERING (Zhang et al., 2013)	allows to chain the middleboxes in the network and steer the traffic at the granularity of the service provider and traffic types based on the policies defined at the controller	Does not address the challenges posed by packet header modification
SDN Service Chaining (Blendin et al., 2014)	Offers dynamic service chain and provides the API to add, remove and modify the service chain	Service instances are isolated in the network; current network status is not considered for service function deployment
Policy based Dynamic Service Chaining (Scheid et al., 2016)	Uses service chaining graphs over policy based management system to achieve the dynamic service chaining in the network	Current network status has not been considered for service chain deployment
SECaaS (Migault et al., 2018)	Enables the SSF in different domains to collaborate for mitigating attacks	Does not have consideration for security rules

Table 6 summarizes different service chaining applications with their main purpose and shortcomings.

4.3. Security policy management

The centralized control plane in the SDN enables to specify the high-level policies at the controller, which can be deployed in the switches dynamically through a southbound API. Low-level implementation details are abstracted from the network administrator. In this section, we examine policy based management system using SDN to protect the network.

Lorenz et al. (2017) summarized how a traditional enterprise network can be extended to integrate the concepts of SDN and NFV. By taking an example of stateful firewalling, they showed three design patterns for the implementation: controller centric, VNF centric and hybrid approach. In addition, the authors also provided the pros and cons of each approach, which can be used as a guideline for the deployment of SDN and NFV appliances in the traditional enterprise networks.

The PolicyCop (Bari et al., 2013) leverages the programmability and logically centralized controller features of SDN to achieve dynamic configuration in their autonomic QoS policy enforcement framework. It offers an interface for specifying QoS policies and leverages the southbound API of SDN controller to enforce policies. The advantages provided by the PolicyCop over the traditional autonomic QoS framework are listed below:

1. It offers per flow control and dynamic flow aggregation in the network contrary to aggregation based on Type of Service (TOS) field in the traditional network.
2. It can specify new traffic classes at run-time without halting the network services.
3. Traditional QoS mechanisms require a number of different protocols for performing routing, MPLS label exchange etc. Differently, PolicyCop can tackle this problem by using OpenFlow protocol for communication between network services and data plane devices.

However, PolicyCop does not provide any grammar to express the high-level policies. It mainly monitors the network and performs the dynamic configuration if the network status changes.

The HACFlow (Rosendo et al., 2017) is an autonomic policy based framework for access control management in SDN networks. It addresses the challenges involving a large set of access control rules, detecting conflicting policies, specifying priorities, delegating rights and reacting to against network status changes. It aims to simplify and automate the network management tasks, allowing network administrators to govern the network by specifying dynamic, fine-grained, and high-level access control policies. In the framework, policies are expressed using Organization Based Access Control model (OrBAC) (Kalam et al., 2003).

The Hierarchical Flow Table (HFT) (Ferguson et al., 2012) framework defines policies in hierarchical way to specify the context, in which network resources can be used among multiple entities. In HFT, policies are expressed in a tree-like structure, and each node in the tree can decide how to perform on each packet. Policy nodes contain a set of policy atoms, which is comprised of match and action pair. The main idea of this work is to resolve the conflicts among the policies rather than translating the high-level policies into low-level rules. To resolve conflicts, it relies on a user defined conflict resolution operator.

The policy refinement toolkit (Machado et al., 2015) allows network administrators to express Service Level Agreements (SLAs) without delving into low-level configuration details of network devices. The main focus is to translate the high-level objectives defined in the SLA into low-level rules. The toolkit automatically translates the high-level policies into low-level rules for deployment in the network devices. The policy refinement can be achieved in two stages: (a) in the first phase, SDN controller gathers the network informations (e.g., latency, jitter) from the data-plane and stores them in the policy repository. (b) In the second stage, the framework translates the high-level SLAs into low-level rules for deployment. Depending on the informations collected in the first phase, the framework provides the best possible configuration for the high-level SLA. The framework offers the best possible configuration for the high-level SLA depending on the information collected from the data plane in the first phase. However, in this framework, it is not clear how the network elements are configured beyond the policy path computation.

Gao et al. (2015) introduced an SDN-based policy deployment framework. It is a three-layer framework that offers a mechanism for policy deployment. The main components of the framework are: (i) policy engine; (ii) job scheduler; and (iii) device manager. The policy engine takes the high-level policies and network topology as inputs and translates the policies into specific low-level device configuration. The job scheduler maintains the real-time configuration of data plane devices. The device manager module is responsible for the deployment of low-level rules in the data plane devices. However, it did not consider the different classes of policies and conflict that can arise in policy enforcement.

The EnforSDN (Ben-Itzhak et al., 2015) architecture offers a mechanism to integrate the network appliances such as firewall, Intrusion Prevention System (IPS), and IDSs into SDN networks. The main purpose is to centralize the policy resolution layer and route the flow through appropriate network appliances such as firewall, IDS, etc. In the architecture, policy resolution layer is decoupled from policy enforcement layer and is centralized at the controller. High-level policies are expressed at the policy resolution layer. An application called *EnforSDN* manager is deployed at the controller, which can be used to express the high level policies. The SDN controller configures the data plane devices to steer the flow through appropriate instances of network appliances in the network. EnforSDN framework can efficiently route the traffic through the network by avoiding the unnecessary processing through the middleboxes.

Tripathy et al. (2016) proposed a policy management framework for policy enforcement after detecting and resolving conflicts by a certi-

Table 7
SDN-based policy management.

Policy Management	Short description	Policy Language
Hierarchical Flow Table (HFT) (Ferguson et al., 2012)	provides a framework to realize the hierarchical policies in the SDN	Policies are defined as a tree in the framework
Policy refinement toolkit (Machado et al., 2015)	facilitates to define the SLA without knowing the implementation details	Controlled natural language is used to define the high-level policies
EnforSDN (Ben-Itzhak et al., 2015)	Decouples policy resolution layer from policy enforcement layer	No policy specification language is given
Computer Network defense Policy (Gao et al., 2015)	provides policy based network management to achieve security in the network	Provides Computer Network Defense Policy (CNDP) policy specification
Policy Management Framework (Tripathy et al., 2016)	Provides security, correctness and adaptability for on-demand changes in the policies and ensures that policies are enforced by a certified server and conflicts are resolved before enforcement	No policy specification language is given
PolicyCop (Bari et al., 2013)	Provides policy-based QoS management using SDN	No policy specification language is given
PbSA (Karmakar et al., 2016)	Specify path and flow based security policies to protect services in SDN	Policies are defined in JSON format
Adaptive Policy Management (Sahay et al., 2017b)	Policy framework to protect the ISP as well as its customers	Policies are defined in CNL format
HACFlow (Rosendo et al., 2017)	Provides an autonomic policy based framework for access control management in SDN networks	Access control policies are defined using OrBAC model

fied server. The authors argue that most of the frameworks (Ben-Itzhak et al., 2015; Machado et al., 2015; Bari et al., 2013; Ferguson et al., 2012; Gao et al., 2015; Porras et al., 2012) could not address the threats that arise from application layer because of compromised server. They also found that these existing frameworks did not consider different classes of policies. Furthermore, there is a lack of consistency checking between the updated policy rules and the existing flow rules. To mitigate this issue, FortNOX introduced a mechanism for conflict resolution, role based authorization, and rule set reduction for NOX controller (Porras et al., 2012). The proposed framework provides three network functions: a) trust verify, b) policy conflict resolve, and c) policy consistency checking. Trust verify function identifies the compromised applications and applies the appropriate measures for control. Policy conflict resolve function can analyze the conflicts among the policies and resolve them for deployment. Policy consistency function checks whether the existing flow table entries in the switches comply with the high-level policies. These functions guarantee security, correctness and adaptability for on-demand changes in the policy rules. The proposed architecture ensures the policies are enforced through certified server.

The PbSA (Karmakar et al., 2016) is a policy based security framework aiming to secure the end services across multiple domains in an SDN environment. The authors describe a policy language to define security and network policies that are important for protecting SDN services and communications. For policy specification, the framework considers variety of attributes associated with users, devices and context information, such as location, routing information, services accessed as well as security attributes associated with the switches and the controllers in different domains. An important aspect of this framework is to specify path and flow based security policies, which are important for protecting end to end services in SDN.

In Sahay et al. (2017b), the authors proposed a policy management architecture for securing end user network. It offers a policy language based on controlled natural language (CNL) (Kuhn, 2014) to express the high-level security and network policies. The CNL based policy language allows to express the policies in human understandable format. The framework enables the multiple customers to share security events with their Internet Service Provider (ISP) for appropriate actions in the ISP network. The security and network policies should consider the current network status to offer reaction policies, like the context of end user and service level agreement (SLA) with end user.

Table 7 provides a short description of SDN-based policy management frameworks and policy language that they can support. Table 7 also shows that only (Ferguson et al., 2012; Machado et al., 2015; Gao et al., 2015) provide a grammar to define the high-level policies.

5. Smart grid security using SDN

Communication infrastructure in smart grid is comprised of heterogeneous devices such as controller, sensors and actuators that exchange real time information for monitoring the status of grid infrastructure. In the existing power grid infrastructure, the communication network relies on the traditional network architecture which makes it static in nature. In this traditional communication infrastructure, it is very difficult to get the real-time network status for self-healing and adaptive resource scheduling. Due to the lack of unified control, it is difficult to implement fine grained traffic monitoring and access control (Zhang et al., 2016). Furthermore, in case of network fault or congestion network operators are required to manually configure the networking devices to adapt the network according to the current status. Particularly, reliance on manual configuration can increase the implementation complexity and time consumption to adapt the network according to requirements. Recently, there have been some research efforts trying to improve the management of communication infrastructure of grid using SDN (Aydeger et al., 2016; da Silva et al., 2015; Nobakht et al., 2016; Gyllstrom et al., 2014).

An SDN-based smart grid infrastructure is shown in Fig. 2. As we can see, an SDN controller in the smart grid architecture provides an additional control layer, which offers a platform to deploy new algorithms and grid applications for spreading the power grid data. It also enables configuring the data plane by deploying rules in switches and routers. As shown in Fig. 2, in the SDN-based smart grid environment, SCADA and SDN controller can collaborate with each other, which can ease the management of the network. SDN controller manages the cyber or network layer of the smart grid infrastructure. Moreover, the SCADA controller analyzes the data received from the power grid layer and reports any anomalous deviation to the SDN controller. This can configure the rule in the switches to mitigate the cyber attacks or to redirect the traffic through another path. Fig. 2 shows that the switches or routers are also connected with the power grid infrastructure. If there is a network failure or congestion in some parts of the infrastructure, then based on the decision from the SCADA controller, the SDN controller can configure the rules in the switches to divert the traffic to another power grid station. Global visibility of the SDN controller offers the detailed traffic monitoring and facilitates the deployment of fail-over communication paths.

Nobakht et al. (2016) proposed an intrusion detection and mitigation framework, called IoT-IDM, by using OpenFlow to protect smart home from network attacks. Detection unit in the IoT-IDM framework examines the network traffic for suspicious and malicious activities. If

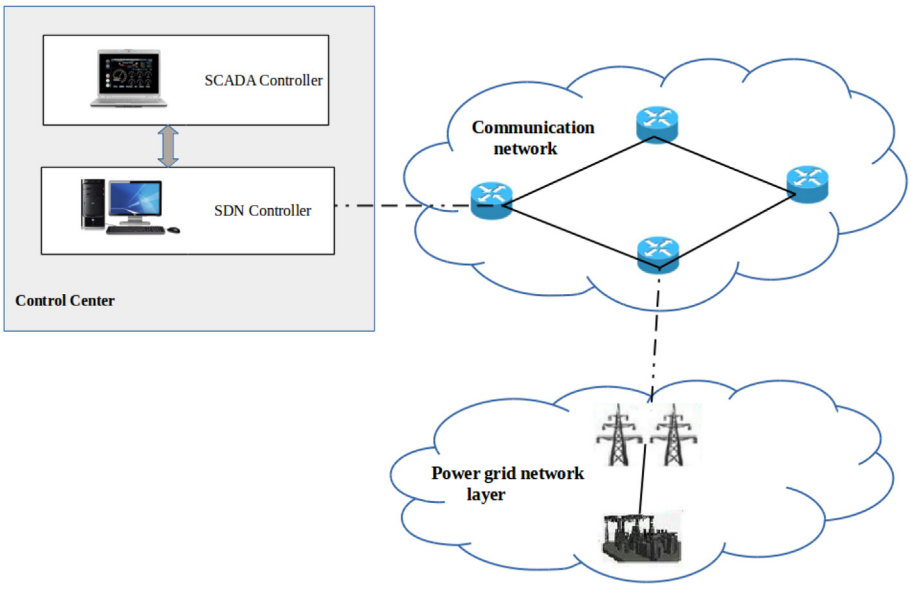

Fig. 2. Smart grid Architecture.

it detects any suspicious or malicious traffic, it can send all the relevant information to the mitigation engine which is responsible for enforcing the countermeasure. Modular architecture of the framework allows to deploy any machine learning algorithms for detecting malicious traffic.

da Silva et al. (2015) presented a mechanism to handle anti-eavesdropping in SCADA systems using SDN. It relies on redundant network connectivity to distribute the traffic across multiple paths. Each path only transmits a portion of the packets towards end hosts to thwart the interception of flows by unauthorized hosts.

In Aydeger et al. (2016), the authors introduced an architecture based on SDN for resilient communication in smart grid network. They proposed to use multiple wireless links for communication which can be used during emergencies. In the architecture, an SDN based communication network is deployed within and among the substations for communication. Each substation maintains an SDN switch which can be configured by the global SDN controller at the control center for forwarding the traffic among different substations. Moreover, each substation has a local SDN controller, which is responsible for controlling the traffic within the substation.

The authors in Gyllstrom et al. (2014), introduced a link failure detection, reporting and recovery mechanism based on SDN for smart grid communication network. They presented a set of algorithms called APLESEED, which could run at the SDN controller with the aim to detect a link failure, compute backup paths and quickly deploy the rules in the data plane. The objective of APLESEED is to compute the optimal path after the link failure in the network. They also presented a local optimization mechanism to create minimal sets of forwarding rules to reduce the forwarding state in the switches.

Bhardwaj et al. (2018) presented an architecture ShadowNet to defend against application-level IoT-DDoS attack by leveraging the edge computing infrastructure. The architecture makes the edge as the first line of defense against IoT-DDoS attack. The edge functions collect the information about the incoming traffic and send those information to a nearby detection service. It expedites the detection and mitigation of such attacks by limiting their impact.

Rubio-Hernan et al. (2018) proposed a cyber physical architecture by leveraging the characteristics of SDN. The architecture combines the control-theoretic solutions together with the SDN technology to tackle the threats to cyber physical system for improving the resilience. The architecture allows cooperation between traditional Feedback Controllers and SDN devices to mitigate related threats.

Aydeger et al. (2015) presented an approach to improve the resilience by utilizing SDN for redundant communications in smart grid environment. If a wired link is failed, then the SDN controller makes the wireless link up. The authors illustrated their approach using mininet with ns-3. In the demo, the authors purposely dropped the wired link and illustrated that the SDN controller could update the flow table to enable the wireless link as a backup path to achieve resilience. This work aims to connect the Mininet (Lantz et al., 2010) with the ns-3 to show the proposed scenario.

The Flow Validator (Kumar and Nicol, 2016) is a framework which leverages the north bound API of SDN to gather information about the current state of the SDN. Depending on this information, Flow Validator performs fast failover to achieve the resilience in smart grid communication.

An SDN based architecture, called ARES, is introduced in Lopes et al. (2017), for autonomic and resilient communication in smart grid. The framework introduced a new API SCADA-NG at the management layer. This API is responsible for controlling, configuring and monitoring all the information in smart grid environment. Whenever a failure occurs, the backup path is automatically chosen. Moreover, the framework presented an algorithm to calculate the shortest path to all possible source-destination pairs in the network. The authors implemented the framework using mininet and RYU SDN controller. In comparison to other proposals, it shows the maximum recovery time of 610 μs, which is a significant improvement.

In Al-Rubaye et al. (2018), the authors introduced an architecture by leveraging the SDN paradigm to improve resilience in the smart grid network. The authors proposed a fault tolerance approach in which the architecture evaluates both the end-to-end delay and the data flow traffic under the presence of faults. Moreover, the authors presented an algorithm, which runs on the SDN controller to compute the path between SDN controller and switches. The experimental results show that the proposed architecture incurs low end-to-end delay in comparison to traditional network settings.

5.1. Some insights and lessons learned

In this section, we have provided a detailed description of mechanisms proposed to secure smart grid environment using SDN tech-

Table 8
SDN-based smart grid infrastructure.

Smart Grid	Main Purpose	Simulation Tool
IoT-IDM (Nobakht et al., 2016)	provides a framework using OpenFlow to protect smart home from network attacks	Floodlight controller used in mininet for implementation
Anti-eavesdropping in SCADA (da Silva et al., 2015)	Distribute the traffic in the network through multiple paths to defend against eavesdropping	POX controller is used in mininet for implementation.
SDN based Inter Substation framework (Aydeger et al., 2016)	Relies on redundant links to achieve resilience	OpenDaylight controller used in mininet for implementation
APPLESSED (Gyllstrom et al., 2014)	Proposed a set of algorithm to detect link failure, computing back up paths, and installation of backup paths	Mininet and POX controller
ShadowNet (Bhardwaj et al., 2018)	Introduced an architecture to defend against application-level IoT-DDoS attack by leveraging the edge computing infrastructure	Created the testbed using GENI platform
Flow Validator (Kumar and Nicol, 2016)	Presented an architecture for fast failover to achieve the resilience in smart grid communication	Mininet
ARES (Lopes et al., 2017)	Main idea is to achieve autonomic, resilient communication and fast failover in smart grid	Mininet
Rubio-Hernan et al. (2018)	Combines the control-theoretic solutions together with the SDN technology to tackle the threats for improving the resilience.	Mininet
Aydeger et al. (2015)	Improves the resilience by utilizing SDN for redundant communications in smart grid environment.	Mininet and NS-3
IIOT (Al-Rubaye et al., 2018)	Improves the resilience by utilizing SDN for redundant communications in smart grid environment.	OpenDaylight controller, Openstack

nology. The literature review reveals that using SDN in smart grid communication infrastructure can help ease the network management tasks. It allows deploying additional functionalities in the communication devices to deal with the failures. It offers flexibility to deploy various algorithms and to maintain redundant paths in the network which can be deployed dynamically. Furthermore, the SDN controller can have a global view of the network and make the routing decisions based on the current status of the network. The SDN controller can configure the flow table of switches dynamically to route the flow through another path when needed. It reduces the time consumption to recover from link failure or congestion in the network. However, the SDN controller can be a single point of failure, therefore, leading to communication breakdown in smart grid environment. First, further study should be required to tackle the failure of SDN controller in the smart grid environment. Moreover, communication between the SDN controller and the cyber-physical controller should be secured for a better collaboration between them. Table 8 summarizes related SDN-based applications that aim to protect and improve the resilience of the smart grid infrastructure using SDN.

6. Research challenges and future directions

In this section, we discuss research challenges and potential future research directions on how to enhance the network security via SDN. In addition, it is worth noting that SDN itself has many security issues that need to be addressed before/when deploying SDN for improving network security.

6.1. SDN security

In the above sections, we have discussed many research studies on improving network security using SDN technology. In this part, we present major security issues of SDN itself, which need to be addressed before the deployment of SDN. There have been some efforts in identifying different attack vectors in SDN (Kreutz et al., 2013), as well as security issues in the SDN enabled network (Alsmadi and Xu, 2015; Shin et al., 2014; Benton et al., 2013; Sezer et al., 2013).

In Kreutz et al. (2013) the authors highlighted seven threat vectors in SDN architecture. While some of the attack vectors are present in the traditional network, the others are more specific to SDN network. For instance, attacks on logically centralized controller and control-data plane communication channel are specific to SDN network. The attack on logically centralized controller poses more serious threat as it impacts the whole network. Forged flows can be used to attack switches and controllers in the network. Moreover, communication channel between data and control plane can also be attacked by the forged traffic. Furthermore, attacker can use the compromised applications on the SDN controller to program the network.

Several countermeasures have been proposed to address the security threats in SDN. For example, packet filtering, rate limiting, rule timeout can be used to defend against DDoS and forged traffic. Rules can also be proactively deployed in the switches to reduce the frequent interaction between the controller and the switches. It protects the controller against control plane saturation attack. Availability of the controllers can be achieved through replicating the controllers (Botelho et al., 2013).

Implementing trust between the applications and the SDN controller is also important, since malicious or poorly configured applications at the controller can compromised the network. Wen et al. (2013) introduced a fine grained permission system as a first line of defense to tackle the attacks launched through northbound API of the controller. They summarized a set of 18 permissions to be enforced on applications for access. Moreover, it is worth mentioning that there are many other proposals of access controls and authentication mechanism for protecting SDN from security threats (Klaedtke et al., 2014; Toseef et al., 2014). Klaedtke et al. (2014) proposed an access control scheme for SDN controller. The proposed access control techniques detect and resolve conflicts close to the southbound API. Moreover, C-BAS (Toseef et al., 2014) provided a certificate-based authentication, authorization and accounting (AAA) mechanism for SDN experimental facilities. These studies are preliminary research efforts in the direction of securing the SDN network; however, there is still a considerable amount of work needs to be done before SDN can be implemented in a practical scenario.

6.2. Compatibility with traditional networking

Network operators might be reluctant to completely replace their traditional IP-based networking infrastructure with SDN technology.

Therefore, SDN should be incrementally deployable to provide the compatibility with the existing networking infrastructure. Compatibility with the legacy infrastructure can be assured by deploying hybrid switches, i.e. switches which can be configured to behave as legacy as well as OpenFlow switches. For instance, some ports on switches can be configured for OpenFlow enabled network while others are assigned for the legacy network. There have been some efforts dedicated to address the compatibility issues of SDN with existing traditional IP based networking (Najd and Shue, 2017; Hong et al., 2016). DeepContext (Najd and Shue, 2017) introduced a host based SDN approach which is compatible with OpenFlow. In this architecture, end hosts are responsible for storing the rules, and it extends Open vSwitch to provide contextual information to host which can assist controllers in decision making. Hong et al. (2016) proposed a mechanism for incremental deployment of SDN in the ISP (Internet Service Provider) and enterprise network. They performed a systematic analysis on which legacy devices should be upgraded to SDN, and how SDN and legacy devices can interoperate to achieve the traffic engineering objectives in the network.

6.3. Scalability

Scalability is one of the major concerns of SDN, in which SDN control and data plane could cause many scalability issues. Switch-to-controller interaction because every first packet of a new flow is forwarded to the controller cause potential performance bottleneck for the controller and switches. In large data center , network controller is required to handle millions of flows per second (Benson et al., 2010). Therefore, processing overhead at the controller as well as at the data plane can cause many performance issues. Therefore, researchers have devoted their efforts to address the challenges posed by the scalability of the controller and the data plane (Yu et al., 2010; Curtis et al., 2011; Phemius et al., 2014; Tootoonchian and Ganjali, 2010; Hartert et al., 2015; Cai et al., 2010; Tootoonchian et al., 2012). Work such as DIFANE (Yu et al., 2010) reduces the burden of controller by keeping most of the traffic in the data plane and selectively forwarding the traffic through intermediate switches which have the rules to process the incoming traffic. The controller only performs tasks such as partitioning the rules over the switches in the network.

An alternative solution is to design scalable controllers like Maestro (Cai et al., 2010) and NOX-MT (Tootoonchian et al., 2012). These high performance controllers can handle millions of new flows per second with an average latency of few microseconds. Moreover, there have been some proposals to improve the scalability by deploying multiple controllers in the network (Hassas Yeganeh and Ganjali, 2012; Levin et al., 2012; Heller et al., 2012). Kandoo (Hassas Yeganeh and Ganjali, 2012) proposed to distribute the controllers physically in the network for improving the scalability. In Heller et al. (2012), authors try to address the controller placement problem. They address the issue that if given a topology, how many controllers are needed and where these controllers should be deployed to achieve the optimum scalability. In Levin et al. (2012), authors address the problem using state exchange between the different controllers deployed in the network. They identified two key exchange points among the controllers and simulate these scenarios in the context of an SDN load balancer application.

Moreover, limitations of flow table entries in OpenFlow switches also raise the scalability problems. It has been explored that limitations of flow table entries can cause the flow table overflow attack (Qian et al., 2016; Qiao et al., 2016). In Qian et al. (2016), authors proposed a rate limiting approach to limit the amount of traffic that a switch can handle. Flow table sharing approach is introduced by authors in Qiao et al. (2016). In this mechanism, if a switch cannot process a flow, then it can forward it to another switch in the network, which has a spare flow table for processing. Moreover, researchers have also proposed to deploy the software switches in the network to handle the problem posed by the flow table limitations of real hardware switches. Nowadays, in the data center network, network operators deploy software switches at the ingress point of the network to process the new flows (Kreutz et al., 2015).

These research efforts have addressed most scalability concerns regarding the data and the control plane of SDN; however, there is still immense amount of work needed to improve the scalability of SDN. In addition, the scalability of the controller will also be impacted by the applications operated on the controller. For example, security and load balancing applications running on the controller need more time to execute than a basic application, which is only responsible for forwarding the network traffic.

6.4. High-level language abstraction

The high-level programming language in SDN should allow performing low-level device configuration. It should allow programmer or network administrator to perform policy changes in the network automatically. There have been some efforts by the researchers to design a good high-level programming language for SDN (Foster et al., 2010; Voellmy et al., 2012, 2013; Voellmy and Hudak, 2011; Bosshart et al., 2014; Yuan et al., 2014; Anderson et al., 2014; Monsanto et al., 2012; Soulé et al., 2013). However, the existing programming languages in SDN offer constructs to handle specific challenges. However, these basic constructs force programmers to write even a basic SDN applications. Therefore, it is still an open issue offering constructs in high-level languages to manage virtualized network functions in an SDN environment. Furthermore, language constructs should ensure to scale the network resources in an elastic and automated way.

6.5. Resilience in SDN

Nowadays, achieving resilience in security and networking is a top priority. Fault or failure in specific components should not impact the availability of the services in the network. The resilience of SDN environment depends on the fault tolerance in the data plane as well as on the control plane. Therefore, achieving resilience in SDN is a difficult task because failures may be caused from different layers as highlighted in Kim et al. (2012). Implementation of SDN in the data centers of Google has shown that it can be resilient (Mandal, 2015). There have been some efforts from research community to address concern of resilience from the control plane perspective (Kempf et al., 2012; Reitblatt et al., 2013; Krishnamurthy et al., 2014; Sharma et al., 2013; Dixit et al., 2013; Panda et al., 2013; Kuźniar et al., 2013). Most of these mechanisms rely on distributed controller approach to achieve resilience in the SDN environment. However, there are security, consistency and scalability challenges which need to be addressed to achieve resilience using distributed controller environment.

Data plane resilience in the SDN technology is also an important issue for the overall resilience of the network. It needs to be addressed for the widespread deployment of SDN. On a similar pattern, few studies address the problem of data plane resilience (Ramos et al., 2013). It uses the idea of packet header space analysis and depends on switches itself to route the traffic through backup path in case of failure in the main path without involving the controller. On the related line in Sgambelluri et al. (2013), the authors proposed to protect individual segment of path in the network without involving the controller.

6.6. SDN and network function virtualization

Network functions virtualization (NFV) is an emerging technology, which yields numerous benefits. For instance, instead of deploying specialized hardware equipments, virtualized network functions (VNF) in

NFV can be realized through virtual machines, which decreases the cost of purchasing specialized hardware equipments. Moreover, Software defined networking and network function virtualization (NFV) are complementary to each other, i.e., they can be both applied to different types of networks (Haleplidis et al., 2014; Cerrato et al., 2014a; Xia et al., 2014; Gember-Jacobson et al., 2014). Recently, there have some initiative towards combining SDN and NFV (Cerrato et al., 2014b). In Wu et al. (2018b), the authors introduced a novel and dynamic control architecture by leveraging the SDN and NFV technology. The topology of the CPS system is controlled by the SDN controller and sensor functions are developed for the CPS using NFV technology. It improves the management of the network and the closed loop collaboration between the cyber side and physical side. However, there are many security challenges in NFV which need to be addressed (Yang and Fung, 2016). For instance, DDoS attack and compromised VNF can cause huge damage to NFV supported network.

6.7. Big data for informed decision in SDN

Recently there have been some initiative to bring the power of big data to SDN specifically to manage the network resources intelligently (Wu et al., 2018c; Using big data for SDN, 2013; Sideris et al., 2016). Wu et al. (2018c) argue that in the large scale network, multiple SDN controllers are able to collaborate effectively to manage the network. The authors proposed a big data analysis based cluster management architecture to manage the logically centralized and physically distributed controllers in the large scale network. Security of this cluster of controllers is also a challenging task. A secure authentication mechanism is proposed to authenticate the data sources from different controllers. Moreover, an ant colony optimization mechanism is used for big data analysis and optimizing the control plane. A framework is introduced in Sideris et al. (2016), for network intelligence by leveraging the network programmability of SDN and big data principles. The framework provides a fault tolerant, scalable and real-time platform for data extraction to manage the entire network. In Using big data for SDN (2013), the author argues to use the big data analysis technique to manage the network resources, which considers the whole network as a resource rather than a collection of points. However, big data and SDN technology, both are in a nascent stage and there are many open issues which need to be addressed. For instance, efficiency of analytic technique as well as security of data storage mechanism in SDN should be considered. Moreover, security of messages exchanged between different domains and scalability of the big data analysis technique in the large scale SDN network should be addressed for the real time integration of analytic technique with the SDN technology.

6.8. SDN and 5G security

Nowadays, SDN is considered to achieve the demand of the 5G mobile network. SDN has the potential to solve the security issues in the mobile core network. The decoupled control plane from the data plane and logically centralized controller in SDN offers great advantages in solving the security challenges in the 5G mobile network. By leveraging the global visibility of the SDN, consistent flow rules can be deployed at the required location to filter attack traffic or redirect the traffic towards middleboxes. In the regard, Liang and Qiu (2016) proposed an SDN based architecture for 5G network. The architecture offers an on-demand security service running in parallel with mobile network domain. It ensures the security service for the mobile consumers. Different types of applications can be programmed to secure the 5G mobile network that can communicate with the SDN controller, which is responsible to install the flow rules to protect the network. In Ahmad et al. (2017), the authors highlighted the security challenges in 5G network and argue that SDN and NFV can offer potential technological solutions for 5G mobile network. However, they also pointed out that the new types of security threats and challenges will be posed by the 5G mobile network with the deployment of these technological solutions. It is worth noting that addressing these security challenges from the design stage will minimize the impact of potential vulnerabilities.

7. Conclusion

In this paper, we surveyed existing research regarding the application of SDN on securing computer networks. Relevant research in SDN community is still in its early stages, but there is immense work has been done to design and develop different applications to ease the burden of network management by means of SDN. We categorized existing work into nine categories: attack detection, vulnerability detection, attack mitigation, dynamic configuration based on SDN, security policy management using SDN, traffic engineering achieved using SDN, traffic monitoring, deployment of middleboxes, and service chaining. Also, we presented some research efforts on securing smart grid infrastructure via SDN. Then, we discuss research challenges and potential future directions, including SDN security, compatibility, and scalability issues. Our survey aims to stimulate more research to investigate how to design a deployable and secure SDN, and how to apply SDN for achieving network security in a practical scenario.

Acknowledgement

This work was partially supported by the Project of Cyber Resilience for the Shipping Industry (CyberShip), through the Danish Maritime Fund and Orients Fund.

References

van Adrichem, N.L.M., Doerr, C., Kuipers, F.A., 2014. Opennetmon: network monitoring in openflow software-defined networks. In: 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1–8, https://doi.org/10.1109/NOMS.2014.6838228.

Agarwal, S., Kodialam, M., Lakshman, T.V., 2013. Traffic engineering in software defined networks. In: 2013 Proceedings IEEE INFOCOM, pp. 2211–2219, https://doi.org/10.1109/INFCOM.2013.6567024.

Ahmad, I., Kumar, T., Liyanage, M., Okwuibe, J., Ylianttila, M., Gurtov, A., 2017. 5g security: analysis of threats and solutions. In: 2017 IEEE Conference on Standards for Communications and Networking (CSCN), pp. 193–199, https://doi.org/10.1109/CSCN.2017.8088621.

Ahn, L.V., Blum, M., Hopper, N.J., Langford, J., 2003. Captcha: using hard ai problems for security. In: Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques, EUROCRYPT'03. Springer-Verlag, Berlin, Heidelberg, pp. 294–311. http://dl.acm.org/citation.cfm?id=1766171.1766196.

Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A., 2010. Hedera: dynamic flow scheduling for data center networks. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI'10. USENIX Association, Berkeley, CA, USA, p. 19.

Al-Rubaye, S., Kadhum, E., Ni, Q., Anpalagan, A., 2018. Industrial internet of things driven by sdn platform for smart grid resiliency. IEEE Internet Things J. 1, https://doi.org/10.1109/JIOT.2017.2734903.

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

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

Anderson, C.J., Foster, N., Guha, A., Jeannin, J.-B., Kozen, D., Schlesinger, C., Walker, D., 2014. Netkat: semantic foundations for networks. SIGPLAN Not. 49 (1), 113–126, https://doi.org/10.1145/2578855.2535862.

Anon. 1. [link]. URL https://www.ibm.com/developerworks/community/blogs/ibmsyssw/entry/sdn_openflow_for_increased_flexibility_improved_performance_and_simplified_operations?lang=en.

Anon. 2. [link]. URL http://h17007.www1.hpe.com/si/en/networking/solutions/technology/sdn/portfolio.aspx#.W2BvMHVubCI.

Anwer, B., Benson, T., Feamster, N., Levin, D., Rexford, J., 2013. A slick control plane for network middleboxes. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. ACM, New York, NY, USA, pp. 147–148, https://doi.org/10.1145/2491185.2491223.

Arbor Networks, 2016. Worldwide Infrastructure Security Report. Tech. rep.. Arbor Networks.
Argyropoulos, C., Kalogeras, D., Androulidakis, G., Maglaris, V., 2012. Paflomon – a slice aware passive flow monitoring framework for openflow enabled experimental facilities. In: 2012 European Workshop on Software Defined Networking, pp. 97–102, https://doi.org/10.1109/EWSDN.2012.13.
Ashraf, J., Latif, S., 2014. Handling intrusion and ddos attacks in software defined networks using machine learning techniques. In: 2014 National Software Engineering Conference, pp. 55–60, https://doi.org/10.1109/NSEC.2014.6998241.
Aydeger, A., Akkaya, K., Uluagac, A.S., 2015. Sdn-based resilience for smart grid communications. In: 2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN), pp. 31–33, https://doi.org/10.1109/NFV-SDN.2015.7387401.
Aydeger, A., Akkaya, K., Cintuglu, M.H., Uluagac, A.S., Mohammed, O., 2016. Software defined networking for resilient communications in smart grid active distribution networks. In: 2016 IEEE International Conference on Communications (ICC), pp. 1–6, https://doi.org/10.1109/ICC.2016.7511049.
Ballard, J.R., Rae, I., Akella, A., 2010. Extensible and scalable network monitoring using opensafe. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10. USENIX Association, Berkeley, CA, USA, p. 8. http://dl.acm.org/citation.cfm?id=1863133.1863141.
Bari, M.F., Chowdhury, S.R., Ahmed, R., Boutaba, R., 2013. Policycop: an autonomic qos policy enforcement framework for software defined networks. In: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1–7, https://doi.org/10.1109/SDN4FNS.2013.6702548.
Bawany, N.Z., Shamsi, J.A., Salah, K., 2017. Ddos attack detection and mitigation using sdn: methods, practices, and solutions. Arabian J. Sci. Eng. 42 (2), 425–441, https://doi.org/10.1007/s13369-017-2414-5.
Ben-Itzhak, Y., Barabash, K., Cohen, R., Levin, A., Raichstein, E., 2015. Enforsdn: network policies enforcement with sdn. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 80–88, https://doi.org/10.1109/INM.2015.7140279.
Benson, T., Akella, A., Maltz, D., 2009a. Unraveling the complexity of network management. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09. USENIX Association, Berkeley, CA, USA, pp. 335–348. http://dl.acm.org/citation.cfm?id=1558977.1559000.
Benson, T., Akella, A., Maltz, D., 2009b. Unraveling the complexity of network management. In: Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09. USENIX Association, pp. 335–348. http://dl.acm.org/citation.cfm?id=1558977.1559000.
Benson, T., Akella, A., Maltz, D.A., 2010. Network traffic characteristics of data centers in the wild. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10. ACM, New York, NY, USA, pp. 267–280, https://doi.org/10.1145/1879141.1879175.
Benton, K., Camp, L.J., Small, C., 2013. Openflow vulnerability assessment. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. ACM, New York, NY, USA, pp. 151–152, https://doi.org/10.1145/2491185.2491222.
Bhardwaj, K., Miranda, J.C., Gavrilovska, A., 2018. Towards iot-ddos prevention using edge computing. In: USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18). USENIX Association, Boston, MA, https://www.usenix.org/conference/hotedge18/presentation/bhardwaj.
Bisong, A., Rahman, S.M., 2011. An Overview of the Security Concerns in Enterprise Cloud Computing. CoRR abs/1101.5613. arXiv:1101.5613, http://arxiv.org/abs/1101.5613.
Blendin, J., Ruckert, J., Leymann, N., Schyguda, G., Hausheer, D., 2014. Position paper: software-defined network service chaining. In: 2014 Third European Workshop on Software Defined Networks, https://doi.org/10.1109/EWSDN.2014.14.
Boite, J., Nardin, P., Rebecchi, F., Bouet, M., Conan, V., 2017. Statesec: stateful monitoring for ddos protection in software defined networks. In: 2017 IEEE Conference on Network Softwarization (NetSoft), pp. 1–9, https://doi.org/10.1109/NETSOFT.2017.8004113.
Bosshart, P., Daly, D., Gibb, G., Izzard, M., McKeown, N., Rexford, J., Schlesinger, C., Talayco, D., Vahdat, A., Varghese, G., Walker, D., 2014. P4: programming protocol-independent packet processors. SIGCOMM Comput. Commun. Rev. 44 (3), 87–95, https://doi.org/10.1145/2656877.2656890.
Botelho, F.A., Ramos, F.M.V., Kreutz, D., Bessani, A.N., 2013. On the feasibility of a consistent and fault-tolerant data store for sdns. In: 2013 Second European Workshop on Software Defined Networks, pp. 38–43.
Braga, R., Mota, E., Passito, A., 2010. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: 35th IEEE Conference on Local Computer Networks (LCN), pp. 408–415, https://doi.org/10.1109/LCN.2010.5735752.
Cai, Z., Cox, A.L., Ng, T.S.E., 2010. Maestro: A System for Scalable Openflow Control.
Cao, Z., Kodialam, M., Lakshman, T.V., 2014. Traffic steering in software defined networks: planning and online routing. SIGCOMM Comput. Commun. Rev. 44 (4), https://doi.org/10.1145/2740070.2627574.
Casado, M., Freedman, M.J., Pettit, J., Luo, J., McKeown, N., Shenker, S., 2007. Ethane: taking control of the enterprise. SIGCOMM Comput. Commun. Rev. 37 (4), 1–12, https://doi.org/10.1145/1282427.1282382.
Cerrato, I., Jungel, T., Palesandro, A., Risso, F., Suñé, M., Woesner, H., 2014a. User-specific network service functions in an sdn-enabled network node. In: 2014 Third European Workshop on Software Defined Networks, pp. 135–136, https://doi.org/10.1109/EWSDN.2014.26.
Cerrato, I., Annarumma, M., Risso, F., 2014b. Supporting fine-grained network functions through intel dpdk. In: 2014 Third European Workshop on Software Defined Networks, pp. 1–6, https://doi.org/10.1109/EWSDN.2014.33.
Chen, X., Mao, Z.M., Van Der Merwe, J., 2009. Shadownet: a platform for rapid and safe network evolution. In: Proceedings of the 2009 Conference on USENIX Annual Technical Conference, USENIX'09. USENIX Association, Berkeley, CA, USA, p. 3. http://dl.acm.org/citation.cfm?id=1855807.1855810.
CISCO, 2018. Tech. rep.. In: Annual Cybersecurity Report. CISCO.
Colville, R.J., Spafford, G., 2010. Configuration Management for Virtual and Cloud Infrastructures. Tech. rep.. Gartner Inc.
Curtis, A.R., Mogul, J.C., Tourrilhes, J., Yalagandula, P., Sharma, P., Banerjee, S., 2011. Devoflow: scaling flow management for high-performance networks. In: Proceedings of the ACM SIGCOMM 2011 Conference, SIGCOMM '11. ACM, New York, NY, USA, pp. 254–265, https://doi.org/10.1145/2018436.2018466.
Dixit, A., Hao, F., Mukherjee, S., Lakshman, T., Kompella, R., 2013. Towards an elastic distributed sdn controller. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. ACM, New York, NY, USA, pp. 7–12, https://doi.org/10.1145/2491185.2491193.
Enns, R., Bjorklund, M., Bierman, A., Schönwälder, J., Jun. 2011. Network Configuration Protocol (NETCONF). RFC 6241. , https://doi.org/10.17487/RFC6241, https://rfc-editor.org/rfc/rfc6241.txt.
F5, 2014. The F5 DDoS Protection Reference Architecture. F5, Tech. rep.
Fayaz, S.K., Tobioka, Y., Sekar, V., Bailey, M., 2015. Bohatei: flexible and elastic ddos defense. In: 24th USENIX Security Symposium (USENIX Security 15). USENIX Association, pp. 817–832.
Fayazbakhsh, S.K., Chiang, L., Sekar, V., Yu, M., Mogul, J.C., 2014. Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags. In: Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, NSDI'14. USENIX Association, Berkeley, CA, USA, pp. 533–546. http://dl.acm.org/citation.cfm?id=2616448.2616497.
Ferguson, A.D., Guha, A., Liang, C., Fonseca, R., Krishnamurthi, S., 2012. Hierarchical policies for software defined networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12. ACM, New York, NY, USA, pp. 37–42, https://doi.org/10.1145/2342441.2342450.
Floodlight Project: Github - Floodlight/oftest: Openflow Switch Test Framework, 2016. http://www.projectfloodlight.org/oftest/.
Foster, N., Freedman, M.J., Harrison, R., Rexford, J., Meola, M.L., Walker, D., 2010. Frenetic: a high-level language for openflow networks. In: Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow, PRESTO '10. ACM, New York, NY, USA, pp. 6:1–6:6, https://doi.org/10.1145/1921151.1921160.
François, J., Dolberg, L., Festor, O., Engel, T., 2014. Network security through software defined networking: a survey. In: Proceedings of the Conference on Principles, Systems and Applications of IP Telecommunications, IPTComm '14. ACM, New York, NY, USA, pp. 6:1–6:8, https://doi.org/10.1145/2670386.2670390.
Gao, J., Xia, C., Wang, S., Zhang, H., 2015. A sdn-based deployment framework for computer network defense policy. In: 2015 4th International Conference on Computer Science and Network Technology (ICCSNT), vol. 01, pp. 1253–1258, https://doi.org/10.1109/ICCSNT.2015.7490959.
Gember-Jacobson, A., Viswanathan, R., Prakash, C., Grandl, R., Khalid, J., Das, S., Akella, A., 2014. Opennf: enabling innovation in network function control. In: Proceedings of the 2014 ACM Conference on SIGCOMM, SIGCOMM '14. ACM, New York, NY, USA, pp. 163–174, https://doi.org/10.1145/2619239.2626313.
Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V., 2014a. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. Comput. Network. 62 (0), 122–136.
Giotis, K., Androulidakis, G., Maglaris, V., 2014b. Leveraging sdn for efficient anomaly detection and mitigation on legacy networks. In: Proceedings of the 2014 Third European Workshop on Software Defined Networks, EWSDN '14. IEEE Computer Society, Washington, DC, USA, pp. 85–90, https://doi.org/10.1109/EWSDN.2014.24.
Github - snrism/florence-dev: Sdn Security Test Framework, 2016. https://github.com/snrism/florence-dev.
Glick, M., Rastegarfar, H., 2017. Scheduling and control in hybrid data centers. In: 2017 IEEE Photonics Society Summer Topical Meeting Series (SUM), pp. 115–116, https://doi.org/10.1109/PHOSST.2017.8012677.
Graur, F., 2017. Dynamic network configuration in the internet of things. In: 2017 5th International Symposium on Digital Forensic and Security (ISDFS), pp. 1–4, https://doi.org/10.1109/ISDFS.2017.7916503.
Gyllstrom, D., Braga, N., Kurose, J., 2014. Recovery from link failures in a smart grid communication network using openflow. In: 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 254–259, https://doi.org/10.1109/SmartGridComm.2014.7007655.
Haleplidis, E., Joachimpillai, D., Salim, J.H., Lopez, D., Martin, J., Pentikousis, K., Denazis, S., Koufopavlou, O., 2014. Forces applicability to sdn-enhanced nfv. In: 2014 Third European Workshop on Software Defined Networks, pp. 43–48, https://doi.org/10.1109/EWSDN.2014.27.
Halpern, J.M., HAAS, R., Doria, A., Dong, L., Wang, W., Khosravi, H.M., Salim, J.H., Gopal, R., Mar. 2010. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810. , https://doi.org/10.17487/RFC5810, https://rfc-editor.org/rfc/rfc5810.txt.
Hartert, R., Vissicchio, S., Schaus, P., Bonaventure, O., Filsfils, C., Telkamp, T., Francois, P., 2015. A declarative and expressive approach to control forwarding paths in carrier-grade networks. SIGCOMM Comput. Commun. Rev. 45 (4), 15–28.
Hassas Yeganeh, S., Ganjali, Y., 2012. Kandoo: a framework for efficient and scalable offloading of control applications. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12. ACM, New York, NY, USA, pp. 19–24, https://doi.org/10.1145/2342441.2342446.

Heller, B., Sherwood, R., McKeown, N., 2012. The controller placement problem. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12. ACM, New York, NY, USA, pp. 7–12, https://doi.org/10.1145/2342441.2342444.

Hong, D.K., Ma, Y., Banerjee, S., Mao, Z.M., 2016. Incremental deployment of sdn in hybrid enterprise and isp networks. In: Proceedings of the Symposium on SDN Research, SOSR '16. ACM, New York, NY, USA, pp. 1:1–1:7, https://doi.org/10.1145/2890955.2890959.

Jero, S., Bu, X., Nita-Rotaru, C., Okhravi, H., Skowyra, R., Fahmy, S., 2017. Beads: automated attack discovery in openflow-based sdn systems. In: Dacier, M., Bailey, M., Polychronakis, M., Antonakakis, M. (Eds.), Research in Attacks, Intrusions, and Defenses. Springer International Publishing, Cham, pp. 311–333.

Kalam, A.A.E., Benferhat, S., Miège, A., Baida, R.E., Cuppens, F., Saurel, C., Balbiani, P., Deswarte, Y., Trouessin, G., 2003. Organization based access control. In: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, POLICY '03. IEEE Computer Society, p. 120. http://dl.acm.org/citation.cfm?id=826036.826869.

Karmakar, K.K., Varadharajan, V., Tupakula, U., Hitchens, M., 2016. A Policy Based Security Architecture for Software Defined Networks. CsCR, https://arxiv.org/abs/1806.02053.

Kempf, J., Bellagamba, E., Kern, A., Jocha, D., Takacs, A., Sköldström, P., 2012. Scalable fault management for openflow. In: 2012 IEEE International Conference on Communications (ICC), pp. 6606–6610, https://doi.org/10.1109/ICC.2012.6364688.

Kim, H., Benson, T., Akella, A., Feamster, N., 2011. The evolution of network configuration: a tale of two campuses. In: Internet Measurement Conference.

Kim, H., Schlansker, M., Santos, J.R., Tourrilhes, J., Turner, Y., Feamster, N., 2012. Coronet: fault tolerance for software defined networks. In: 2012 20th IEEE International Conference on Network Protocols (ICNP), pp. 1–2, https://doi.org/10.1109/ICNP.2012.6459938.

Klaedtke, F., Karame, G.O., Bifulco, R., Cui, H., 2014. Access control for sdn controllers. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN '14. ACM, New York, NY, USA, pp. 219–220, https://doi.org/10.1145/2620728.2620773.

Kreutz, D., Ramos, F.M., Verissimo, P., 2013. Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. ACM, New York, NY, USA, pp. 55–60, https://doi.org/10.1145/2491185.2491199.

Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S., 2015. Software-defined networking: a comprehensive survey. Proc. IEEE 103 (1), 14–76.

Krishnamurthy, A., Chandrabose, S.P., Gember-Jacobson, A., 2014. Pratyaastha: an efficient elastic distributed sdn control plane. In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, HotSDN '14. ACM, New York, NY, USA, pp. 133–138, https://doi.org/10.1145/2620728.2620748.

Krishnan, R., Durrani, M., 2014. Real-time SDN Analytics for DDoS Mitigation.

Kuhn, T., 2014. A survey and classification of controlled natural languages. Comput. Linguist. 40 (1), 121–170, https://doi.org/10.1162/COLI_a_00168.

Kumar, R., Nicol, D.M., 2016. Validating resiliency in software defined networks for smart grids. In: 2016 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 441–446, https://doi.org/10.1109/SmartGridComm.2016.7778801.

Kuźniar, M., Perešíni, P., Vasić, N., Canini, M., Kostić, D., 2013. Automatic failure recovery for software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. ACM, New York, NY, USA, pp. 159–160, https://doi.org/10.1145/2491185.2491218.

Lantz, B., Heller, B., McKeown, N., 2010. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, Hotnets-IX. ACM, New York, NY, USA, pp. 19:1–19:6, https://doi.org/10.1145/1868447.1868466.

Lee, S., Kim, J., Shin, S., Porras, P., Yegneswaran, V., 2017a. Athena: a framework for scalable anomaly detection in software-defined networks. In: 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 249–260, https://doi.org/10.1109/DSN.2017.42.

Lee, S., Yoon, C., Lee, C., Shin, S., Yegneswaran, V., Porras, P.A., 2017b. Delta: a security assessment framework for software-defined networks. In: NDSS.

Levin, D., Wundsam, A., Heller, B., Handigol, N., Feldmann, A., 2012. Logically centralized?: state distribution trade-offs in software defined networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12. ACM, New York, NY, USA, pp. 1–6, https://doi.org/10.1145/2342441.2342443.

Li, J., Berg, S., Zhang, M., Reiher, P., Wei, T., 2014. Drawbridge: software-defined ddos-resistant traffic engineering. In: Proceedings of the 2014 ACM Conference on SIGCOMM. ACM, New York, NY, USA, pp. 591–592, https://doi.org/10.1145/2619239.2631469.

Li, W., Meng, W., Kwok, L.F., 2016. A survey on openflow-based software defined networks. J. Netw. Comput. Appl. 68 (C), 126–139.

Liang, X., Qiu, X., 2016. A software defined security architecture for sdn-based 5g network. In: 2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), pp. 17–21, https://doi.org/10.1109/ICNIDC.2016.7974528.

Lim, S., Ha, J., Kim, H., Kim, Y., Yang, S., 2014. A sdn-oriented ddos blocking scheme for botnet-based attacks. In: 2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 63–68, https://doi.org/10.1109/ICUFN.2014.6876752.

Lopes, Y., Fernandes, N.C., Muchaluat-Saade, D.C., Obraczka, K., 2017. Ares: an autonomic and resilient framework for smart grids. In: 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 222–229, https://doi.org/10.23919/INM.2017.7987283.

Lorenz, C., Hock, D., Scherer, J., Durner, R., Kellerer, W., Gebert, S., Gray, N., Zinner, T., Tran-Gia, P., 2017. An sdn/nfv-enabled enterprise network architecture offering fine-grained security policy enforcement. IEEE Commun. Mag. 55 (3), 217–223, https://doi.org/10.1109/MCOM.2017.1600414CM.

Machado, C.C., Wickboldt, J.A., Granville, L.Z., Schaeffer-Filho, A., 2015. Policy authoring for software-defined networking management. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 216–224, https://doi.org/10.1109/INM.2015.7140295.

Mahimkar, A., Dange, J., Shmatikov, V., Vin, H., Zhang, Y., 2007. Dfence: transparent network-based denial of service mitigation. In: Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation, NSDI'07. USENIX Association, Berkeley, CA, USA, p. 24. http://dl.acm.org/citation.cfm?id=1973430.1973454.

Mandal, S., 2015. Experience with B4: Google's Private SDN Backbone. USENIX Association, Santa Clara, CA.

Mann, V., Vishnoi, A., Kannan, K., Kalyanaraman, S., 2012. Crossroads: seamless vm mobility across data centers through software defined networking. In: 2012 IEEE Network Operations and Management Symposium, pp. 88–96, https://doi.org/10.1109/NOMS.2012.6211886.

McGillicuddy, S., 2013. Radware Adds Open Source DDoS Protection to OpenDaylight Project. Tech. rep.. Radware.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008a. Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38 (2), 69–74.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008b. Openflow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38 (2), 69–74, https://doi.org/10.1145/1355734.1355746.

Mehdi, S.A., Khalid, J., Khayam, S.A., 2011a. Revisiting traffic anomaly detection using software defined networking. In: Sommer, R., Balzarotti, D., Maier, G. (Eds.), Recent Advances in Intrusion Detection. Vol. 6961 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 161–180, https://doi.org/10.1007/978-3-642-23644-0_9.

Mehdi, S.A., Khalid, J., Khayam, S.A., 2011b. Revisiting Traffic Anomaly Detection Using Software Defined Networking. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 161–180, https://doi.org/10.1007/978-3-642-23644-0_9.

Meng, W., Choo, K.R., Furnell, S., Vasilakos, A.V., Probst, C.W., 2018. Towards bayesian-based trust management for insider attacks in healthcare software-defined networks. IEEE Trans. Netw. Serv. Manag. 15 (2), 761–773.

Migault, D., Simplicio, M.A., Barros, B.M., Pourzandi, M., Almeida, T.R., Andrade, E.R., Carvalho, T.C., 2018. A framework for enabling security services collaboration across multiple domains. Comput. Electr. Eng. 69, 224–239. http://www.sciencedirect.com/science/article/pii/S0045790617311242.

Monsanto, C., Foster, N., Harrison, R., Walker, D., 2012. A compiler and run-time system for network programming languages. SIGPLAN Not. 47 (1), 217–230, https://doi.org/10.1145/2103621.2103685.

Najd, M.E., Shue, C.A., 2017. Deepcontext: an openflow-compatible, host-based sdn for enterprise networks. In: 2017 IEEE 42nd Conference on Local Computer Networks (LCN), pp. 112–119, https://doi.org/10.1109/LCN.2017.12.

Naous, J., Gibb, G., Bolouki, S., McKeown, N., 2008. Netfpga: reusable router architecture for experimental research. In: Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow, PRESTO '08. ACM, New York, NY, USA, pp. 1–7, https://doi.org/10.1145/1397718.1397720.

Niyaz, Q., Sun, W., Javaid, A.Y., 2017. A deep learning based ddos detection system in software-defined networking (sdn). ICST Trans. Secur. Saf. 4, e2.

Nobakht, M., Sivaraman, V., Boreli, R., 2016. A host-based intrusion detection and mitigation framework for smart home iot using openflow. In: 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 147–156, https://doi.org/10.1109/ARES.2016.64.

Open networking foundation. URL https://www.opennetworking.org/.

Open Networking Foundation, 2008. What's behind Network Downtime? Tech. rep.. Juniper Networks.

Open Networking Foundation, 2012. SDN Security Considerations in the Data Center. Tech. rep.. ONF.

Open Networking Foundation, 2013. SDN Security Considerations in the Data Center. Tech. rep.. ONF.

Openflow in Europe: Linking infrastructure and applications. URL http://www.fp7-ofelia.eu.

Openflow network research center. URL http://onrc.stanford.edu/.

Panda, A., Scott, C., Ghodsi, A., Koponen, T., Shenker, S., 2013. Cap for networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. ACM, New York, NY, USA, pp. 91–96, https://doi.org/10.1145/2491185.2491186.

Peng, H., Sun, Z., Zhao, X., Tan, S., Sun, Z., 2018. A detection method for anomaly flow in software defined network. IEEE Access 6, 27809–27817, https://doi.org/10.1109/ACCESS.2018.2839684.

Phemius, K., Bouet, M., Leguay, J., 2014. Disco: distributed multi-domain sdn controllers. In: 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1–4, https://doi.org/10.1109/NOMS.2014.6838330.
Porras, P., Shin, S., Yegneswaran, V., Fong, M., Tyson, M., Gu, G., 2012. A security enforcement kernel for openflow networks. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12. ACM, New York, NY, USA, pp. 121–126, https://doi.org/10.1145/2342441.2342466.
Qazi, Z.A., Tu, C.-C., Chiang, L., Miao, R., Sekar, V., Yu, M., 2013. SIMPLE-fying middlebox policy enforcement using SDN. SIGCOMM Comput. Commun. Rev. 43 (4), 27–38, https://doi.org/10.1145/2534169.2486022.
Qian, Y., You, W., Qian, K., 2016. Openflow flow table overflow attacks and countermeasures. In: 2016 European Conference on Networks and Communications (EuCNC), pp. 205–209, https://doi.org/10.1109/EuCNC.2016.7561033.
Qiao, S., Hu, C., Guan, X., Zou, J., 2016. Taming the flow table overflow in openflow switch. In: Proceedings of the 2016 ACM SIGCOMM Conference, SIGCOMM '16. ACM, New York, NY, USA, pp. 591–592, https://doi.org/10.1145/2934872.2959063.
Ramos, R.M., Martinello, M., Rothenberg, C.E., 2013. Slickflow: resilient source routing in data center networks unlocked by openflow. In: 38th Annual IEEE Conference on Local Computer Networks, pp. 606–613, https://doi.org/10.1109/LCN.2013.6761297.
Reitblatt, M., Canini, M., Guha, A., Foster, N., 2013. Fattire: declarative fault tolerance for software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. ACM, New York, NY, USA, pp. 109–114, https://doi.org/10.1145/2491185.2491187.
Rosendo, D., Endo, P.T., Sadok, D., Kelner, J., 2017. An autonomic and policy-based authorization framework for openflow networks. In: 2017 13th International Conference on Network and Service Management (CNSM), pp. 1–5, https://doi.org/10.23919/CNSM.2017.8255990.
Rubio-Hernan, J., Sahay, R., De Cicco, L., Garcia-Alfaro, J., 2018. Cyber-physical architecture assisted by programmable networking. Internet Technol. Lett. 1 (4), e44, https://doi.org/10.1002/itl2.44, https://onlinelibrary.wiley.com/doi/pdf/10.1002/itl2.44.
Sahay, R., Blanc, G., Zhang, Z., Debar, H., 2015. Towards autonomic ddos mitigation using software defined networking. In: Proceedings of the NDSS Workshop on Security of Emerging Technologies (SENT).
Sahay, R., Blanc, G., Zhang, Z., Debar, H., 2017a. Aroma: an sdn based autonomic ddos mitigation framework. Comput. Secur. 70, 482–499. https://doi.org/10.1016/j.cose.2017.07.008.
Sahay, R., Blanc, G., Zhang, Z., Toumi, K., Debar, H., 2017b. Adaptive policy-driven attack mitigation in sdn. In: Proceedings of the 1st International Workshop on Security and Dependability of Multi-Domain Infrastructures, XDOMO'17. ACM, New York, NY, USA, pp. 4:1–4:6, https://doi.org/10.1145/3071064.3071068.
Scheid, E.J., Machado, C.C., dos Santos, R.L., Schaeffer-Filho, A.E., Granville, L.Z., 2016. Policy-based dynamic service chaining in network functions virtualization. In: 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 340–345, https://doi.org/10.1109/ISCC.2016.7543763.
Scott, C., Wundsam, A., Raghavan, B., Panda, A., Or, A., Lai, J., Huang, E., Liu, Z., El-Hassany, A., Whitlock, S., Acharya, H., Zarifis, K., Shenker, S., 2014. Troubleshooting blackbox sdn control software with minimal causal sequences. SIGCOMM Comput. Commun. Rev. 44 (4), 395–406, https://doi.org/10.1145/2740070.2626304.
Scott-Hayward, S., O'Callaghan, G., Sezer, S., 2013. SDN security: a survey. In: 2013 IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1–7.
Sezer, S., Scott-Hayward, S., Chouhan, P.K., Fraser, B., Lake, D., Finnegan, J., Viljoen, N., Miller, M., Rao, N., 2013. Are we ready for sdn? implementation challenges for software-defined networks. IEEE Commun. Mag. 51 (7), 36–43.
Sgambelluri, A., Giorgetti, A., Cugini, F., Paolucci, F., Castoldi, P., 2013. Openflow-based segment protection in ethernet networks. IEEE/OSA J. Opt. Commun. Netw. 5 (9), 1066–1075, https://doi.org/10.1364/JOCN.5.001066.
Sharma, S., Staessens, D., Colle, D., Pickavet, M., Demeester, P., 2013. Openflow: meeting carrier-grade recovery requirements. Comput. Commun. 36 (6), 656–665, https://doi.org/10.1016/j.comcom.2012.09.011.
Shin, S., Porras, P.A., Yegneswaran, V., Fong, M.W., Gu, G., Tyson, M., 2013. FRESCO: modular composable security services for software-defined networks. In: Proceedings of the ISOC Network and Distributed System Security Symposium (NDSS).
Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., Yegneswaran, V., Noh, J., Kang, B.B., 2014. Rosemary: a robust, secure, and high-performance network operating system. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, New York, NY, USA, pp. 78–89, https://doi.org/10.1145/2660267.2660353.
Shu, Z., Wan, J., Lin, J., Wang, S., Li, D., Rho, S., Yang, C., 2016. Traffic engineering in software-defined networking: measurement and management. IEEE Access 4, 3246–3256, https://doi.org/10.1109/ACCESS.2016.2582748.
Sideris, K., Nejabati, R., Simeonidou, D., 2016. Seer: empowering software defined networking with data analytics. In: 2016 15th International Conference on Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), pp. 181–188, https://doi.org/10.1109/IUCC-CSS.2016.033.
da Silva, E.G., Knob, L.A.D., Wickboldt, J.A., Gaspary, L.P., Granville, L.Z., Schaeffer-Filho, A., 2015. Capitalizing on sdn-based scada systems: an anti-eavesdropping case-study. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 165–173, https://doi.org/10.1109/INM.2015.7140289.
da Silva, A.S., Wickboldt, J.A., Granville, L.Z., Schaeffer-Filho, A., 2016. Atlantic: a framework for anomaly traffic detection, classification, and mitigation in sdn. In: NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, pp. 27–35, https://doi.org/10.1109/NOMS.2016.7502793.
Song, C., Park, Y., Golani, K., Kim, Y., Bhatt, K., Goswami, K., 2017. Machine-learning based threat-aware system in software defined networks. In: 2017 26th International Conference on Computer Communication and Networks (ICCCN), pp. 1–9, https://doi.org/10.1109/ICCCN.2017.8038436.
Soulé, R., Basu, S., Kleinberg, R., Sirer, E.G., Foster, N., 2013. Managing the network with merlin. In: Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks, HotNets-XII. ACM, New York, NY, USA, pp. 24:1–24:7, https://doi.org/10.1145/2535771.2535792.
Symantec, 2018. Internet Security Threat Report. Tech. rep.. Symantec.
Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M., 2016. Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258–263, https://doi.org/10.1109/WINCOM.2016.7777224.
Tootoonchian, A., Ganjali, Y., 2010. Hyperflow: a distributed control plane for openflow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN'10. USENIX Association, Berkeley, CA, USA, p. 3.
Tootoonchian, A., Ghobadi, M., Ganjali, Y., 2010. OpenTM: Traffic Matrix Estimator for OpenFlow Networks. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 201–210, https://doi.org/10.1007/978-3-642-12334-4_21.
Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., Sherwood, R., 2012. On controller performance in software-defined networks. In: Presented as Part of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. USENIX, San Jose, CA.
Toseef, U., Zaalouk, A., Rothe, T., Broadbent, M., Pentikousis, K., 2014. C-bas: certificate-based aaa for sdn experimental facilities. In: 2014 Third European Workshop on Software Defined Networks, pp. 91–96, https://doi.org/10.1109/EWSDN.2014.41.
Tripathy, B.K., Sethy, A.G., Bera, P., Rahman, M.A., 2016. A novel secure and efficient policy management framework for software defined network. In: 2016 IEEE 40th Annual Computer Software and Applications Conference, vol. 2, pp. 423–430, https://doi.org/10.1109/COMPSAC.2016.31.
Using Big Data for SDN: How Analytics Will Enable Programmability, 2013. https://searchnetworking.techtarget.com/opinion/Using-big-data-for-SDN-How-analytics-will-enable-programmability.
Voellmy, A., Hudak, P., 2011. Nettle: taking the sting out of programming network routers. In: Proceedings of the 13th International Conference on Practical Aspects of Declarative Languages, PADL'11. Springer-Verlag, Berlin, Heidelberg, pp. 235–249. http://dl.acm.org/citation.cfm?id=1946313.1946339.
Voellmy, A., Kim, H., Feamster, N., 2012. Procera: a language for high-level reactive network control. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, HotSDN '12. ACM, New York, NY, USA, pp. 43–48, https://doi.org/10.1145/2342441.2342451.
Voellmy, A., Wang, J., Yang, Y.R., Ford, B., Hudak, P., 2013. Maple: simplifying sdn programming using algorithmic policies. SIGCOMM Comput. Commun. Rev. 43 (4), 87–98, https://doi.org/10.1145/2534169.2486030.
Wang, Y., Zhang, Y., Singh, V., Lumezanu, C., Jiang, G., 2013. Netfuse: short-circuiting traffic surges in the cloud. In: Communications (ICC), 2013 IEEE International Conference on, pp. 3514–3518.
Wang, R., Jia, Z., Ju, L., 2015. An entropy-based distributed ddos detection mechanism in software-defined networking. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 1, pp. 310–317, https://doi.org/10.1109/Trustcom.2015.389.
Wen, X., Chen, Y., Hu, C., Shi, C., Wang, Y., 2013. Towards a secure controller platform for openflow applications. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13. ACM, New York, NY, USA, pp. 171–172.
Wichtlhuber, M., Reinecke, R., Hausheer, D., 2015. An sdn-based cdn/isp collaboration architecture for managing high-volume flows. IEEE Trans. Netw. Serv. Manag. 12 (1), 48–60.
Wu, J., Dong, M., Ota, K., Li, J., Guan, Z., 2018a. Fcss: fog computing based content-aware filtering for security services in information centric social networks. IEEE Trans. Emerg. Top. Comput. 1, https://doi.org/10.1109/TETC.2017.2747158.
Wu, J., Luo, S., Wang, S., Wang, H., 2018b. Nles: a novel lifetime extension scheme for safety-critical cyber-physical systems using sdn and nfv. IEEE Internet Things J. 1, https://doi.org/10.1109/JIOT.2018.2870294.
Wu, J., Dong, M., Ota, K., Li, J., Guan, Z., 2018c. Big data analysis-based secure cluster management for optimized control plane in software-defined networks. IEEE Trans. Netw. Serv. Manag. 15 (1), 27–38, https://doi.org/10.1109/TNSM.2018.2799000.
Xia, M., Shirazipour, M., Zhang, Y., Green, H., Takacs, A., 2014. SDN and optical flow steering for network function virtualization. In: Presented as Part of the Open Networking Summit 2014 (ONS 2014). USENIX, Santa Clara, CA, https://www.usenix.org/conference/ons2014/technical-sessions/presentation/xia.
Xie, J., Yu, F.R., Huang, T., Xie, R., Liu, J., Liu, Y., 2018. A survey of machine learning techniques applied to software defined networking (sdn): research issues and challenges. IEEE Commun. Surv. Tutor. 1, https://doi.org/10.1109/COMST.2018.2866942.
Yang, W., Fung, C., 2016. A survey on security in network functions virtualization. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft), pp. 15–19, https://doi.org/10.1109/NETSOFT.2016.7502434.

Yao, J., Wang, Z., Yin, X., Shiyz, X., Wu, J., 2014. Formal modeling and systematic black-box testing of sdn data plane. In: 2014 IEEE 22nd International Conference on Network Protocols, pp. 179–190, https://doi.org/10.1109/ICNP.2014.37.

Yap, K.-K., Kobayashi, M., Underhill, D., Seetharaman, S., Kazemian, P., McKeown, N., 2009. The stanford openroads deployment. In: Proceedings of the 4th ACM International Workshop on Experimental Evaluation and Characterization, WINTECH '09. ACM, New York, NY, USA, pp. 59–66, https://doi.org/10.1145/1614293.1614304.

Yu, M., Rexford, J., Freedman, M.J., Wang, J., 2010. Scalable flow-based networking with difane. SIGCOMM Comput. Commun. Rev. 41 (4).

Yu, M., Jose, L., Miao, R., 2013. Software defined traffic measurement with opensketch. In: Presented as Part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13). USENIX, Lombard, IL, pp. 29–42. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/yu.

Yu, M., Zhang, Y., Mirkovic, J., Alwabel, A., 2014. SENSS: software defined security service. In: Presented as Part of the Open Networking Summit 2014 (ONS 2014). USENIX, Santa Clara, CA, https://www.usenix.org/conference/ons2014/technical-sessions/presentation/yu.

Yuan, Y., Alur, R., Loo, B.T., 2014. Netegg: programming network policies by examples. In: Proceedings of the 13th ACM Workshop on Hot Topics in Networks, HotNets-XIII. ACM, New York, NY, USA, pp. 20:1–20:7, https://doi.org/10.1145/2670518.2673879.

YuHunag, C., MinChi, T., YaoTing, C., YuChieh, C., YanRen, C., 2010. A novel design for future on-demand service and security. In: 2010 IEEE 12th International Conference on Communication Technology, pp. 385–388, https://doi.org/10.1109/ICCT.2010.5689156.

Zhang, Y., Beheshti, N., Beliveau, L., Lefebvre, G., Manghirmalani, R., Mishra, R., Patneyt, R., Shirazipour, M., Subrahmaniam, R., Truchan, C., Tatipamula, M., 2013. Steering: a software-defined networking for inline service chaining. In: 2013 21st IEEE International Conference on Network Protocols (ICNP), pp. 1–10.

Zhang, X., Wei, K., Guo, L., Hou, W., Wu, J., 2016. Sdn-based resilience solutions for smart grids. In: 2016 International Conference on Software Networking (ICSN), pp. 1–5, https://doi.org/10.1109/ICSN.2016.7501931.

Zhong, H., Fang, Y., Cui, J., 2017. Lbbsrt: an efficient sdn load balancing scheme based on server response time. Future Gener. Comput. Syst. 68, 183–190. https://doi.org/10.1016/j.future.2016.10.001.

Rishikesh Sahay is a Postdoctoral researcher at Technical University of Denmark. He completed his PhD from Télécom SudParis and University of Pierre and Marie Curie (UPMC) in France. Currently, he is working on cyber resilience for shipping industry. His PhD thesis was focused on autonomic cyber defense using software-defined networking. His research interests include autonomic cyber defense, policy-based network management, software-defined networking, cyber resilience, and network security.

Weizhi Meng is currently an assistant professor in the Cyber Security Section, Department of Applied Mathematics and Computer Science, Technical University of Denmark (DTU), Denmark. He obtained his Ph.D. degree in Computer Science from the City University of Hong Kong (CityU), Hong Kong. Prior to joining DTU, he worked as a research scientist in Infocomm Security (ICS) Department, Institute for Infocomm Research, A*Star, Singapore, and as a senior research associate in CS Department, CityU. He won the Outstanding Academic Performance Award during his doctoral study, and is a recipient of the Hong Kong Institution of Engineers (HKIE) Outstanding Paper Award for Young Engineers/Researchers in both 2014 and 2017. He is also a recipient of Best Paper Award from ISPEC 2018, and Best Student Paper Award from NSS 2016. His primary research interests are cyber security and intelligent technology in security, including intrusion detection, smartphone security, biometric authentication, HCI security, trust management, blockchain in security, and malware analysis. He served as program committee members for 20 + international conferences. He is a co-PC chair for IEEE Blockchain 2018, IEEE ATC 2019, IFIPTM 2019, Socialsec 2019. He also served as guest editor for FGCS, JISA, Sensors, CAEE, IJDSN, SCN, WCMC, etc.

Christian D. Jensen is an Associate Professor and the Head of the Cyber Security section at the Department of Applied Mathematics and Computer Science, Technical University of Denmark. He holds an M.Sc. in Computer Science from the University of Copenhagen and a Ph.D. in Computer Science from Université Joseph Fourier (Grenoble I, France). He held a position as Lecturer in Computer science at Trinity College Dublin from 1998 to 2002, where he was appointed to his current position. He conducts research in the area of security in distributed systems, where he is particularly interested in the development of models, policies and mechanisms that support secure collaboration in open distributed systems, such as pervasive computing, mobile computing and sensor networks. He has published more than 60 peer-reviewed papers in international journals, conferences and workshops.