A Survey on Software-Defined Networking Security

Shanshan Bian, Peng Zhang
State Key Lab on Integrated Services Networks
Xidian University
Xi'an, China
17802929727@163.com
pengzhangzhang@gmail.com

Zheng Yan
State Key Lab on Integrated Services Networks
Xidian University
Xi'an, China
Department of Comnet, Aalto University, Finland
zyan@xidian.edu.cn

ABSTRACT
Software-Defined Networking (SDN) has gained special attention in both academia and industry. It is a new network architecture framework for networking, which decouples the network control plane from the data plane at physical topology. SDN promotes centralization of network control and introduces the ability to program the network. However, the development of the SDN is constrained by various security concerns, e.g., the central management of SDN is ideal for security attack. In this paper, we investigate potential security threats and specify security requirements accordingly. Existing security solutions in SDN are seriously reviewed and evaluated based on the specified security requirements in order to figure out open issues and motivate future research efforts.

Keywords
Software-Defined Networking (SDN); Security; Architecture.

1. INTRODUCTION
The Internet is based on heterogeneous network structures with very complicated networking protocols. The management of such networks is becoming more and more difficult and inflexible to make any changes. At the same time, new services continuously emerge while the Internet traffic has grown rapidly with enormous user demands. All of above increase the cost on network operation and maintenance. For satisfying the new requirements appeared in markets, network architecture requires innovation.

Software Defined Networking (SDN) is an emerging network paradigm that enables network administrators to configure network resources very quickly and adjust network-wide traffic flows to meet changing needs dynamically. The term SDN was originally coined to represent the ideas and work around OpenFlow [11] at Stanford University, USA [40]. As originally defined, SDN refers to a network architecture where the forwarding state in the data plane is managed by a remotely controlled plane decoupled from the former. The networking industry has shifted from this original view of SDN by referring to anything that involves software as being SDN on many occasions.

As a promising technology, SDN have drawn considerable attentions in academia and industry in recent years. In academia, many researchers have paid their attentions to SDN taxonomy, requirements, architecture and security. In industry, Google has deployed an SDN to interconnect its data centers across the globe [34]. Many big IT companies have joined SDN consortia such as the ONF and the OpenDaylight initiative [30], which shows the importance of SDN from an industrial perspective. There are a number of on-going initiatives on SDN standardization, e.g., 3GPP and ETSI. The adoption of SDN often spans over different areas of IT and networking.

SDN is a network architecture framework that decouples data forwarding and control planes. According to the SDN architecture established by the Open Networking Foundation (ONF) [30], SDN controllers act as entities that manage the control plane and offer the possibility for the SDN application to further extend the functionality of a network. On the other hand, the novel structure of SDN also introduces potential security vulnerabilities, e.g., the centralized controller is ideal for security attack. In case the control plane is hijacked, the whole network is out of control. Therefore, security is one key issue to be addressed before adopting SDN into real use.

The contributions of this paper are summarized as below.

- We investigate the security issues of SDN and specify security requirements accordingly.
- We review the literature about SDN security countermeasures by analyzing and comparing their advantages and disadvantages according to the security requirements.
- We further figure out a number of open issues and propose future research directions to motivate SDN security research.

The rest of the paper is organized as follows. Section 2 briefly introduces system architecture of SDN. Particularly, we introduce its main technologies to highlight their specific characteristics. Section 3 explores SDN architecture, investigates security threats and specifies security requirements accordingly. In section 4, we comprehensively review the security countermeasures of SDN by applying the architecture and requirements as a measure to analyze their effectiveness and comprehensiveness. Furthermore, we discuss open research issues and propose future research directions in Section 5. Finally, a conclusion is presented in the last section.

2. OVERVIEW OF SDN
2.1 System Architecture
In 2006 SANE [1] architecture was firstly proposed. This architecture was based on a logically centralized controller responsible for authentication of hosts and policy enforcement. Ethane [9] extended the work of SANE but used another approach,

MOBIMEDIA 2016, June 18-19, Xi'an, People's Republic of China

DOI 10.4108/eai.18-6-2016.2264176

which required less alteration to the original work. It controlled the work through the use of two components: a centralized controller responsible for enforcing global policy and Ethane switches that forwarded packets based on the flow table. This simplified network control allowed the data and control plane to be separated to allow for more programmability.

SDN is a new software based network architecture and its largest special envoy is to loose coupling between control plane and data plane. The basic architecture of SDN consists of three layers: application plane, control plane, data plane and the interfaces between them as shown in Figure 1.

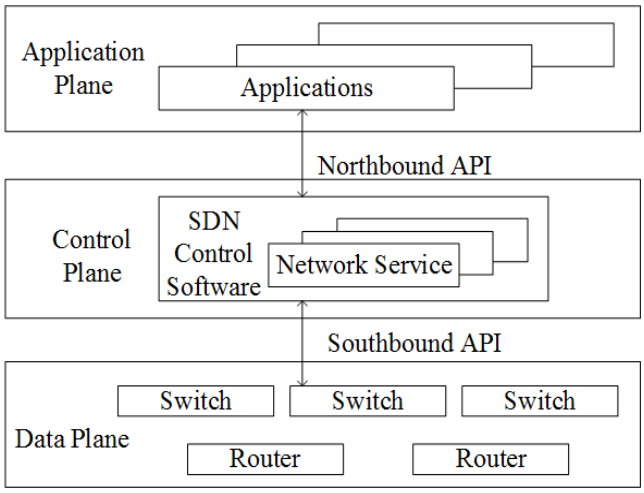

Figure 1. SDN Architecture

The application plane is also called as the application layer, which includes a variety of services and applications such as Deep Packet Inspector (DPI), Intrusion Detection Systems (IDSs), Intrusion Prevention Systems (IPSs), security monitors, load balancers and access controllers.

The control plane is also called as the control layer of SDN. It is responsible for unified control of the underlying forwarding devices, as well as the upper level of applications to provide network capability calls. The control plane utilizes technologies such as link discovery, topology management, policy formulation, table entry and so on.

The data plane is also called the underlying forwarding layer of SDN. This layer mainly comprises the forwarding devices such as routers and switches. The forwarding devices forward data packets based on flow tables managed by the control plane. The data plane forwards data flows, collects network information, and sends network statues to the control plane.

The southbound interface enables communications between the control plane and the data plane. The control plane and the application plane communicate through the northbound interface.

Based on this basic architecture, a number of security analyses have been recently been performed, which have found that the altered elements or relationship between the elements in the SDN framework introduce new vulnerabilities, which were not present before SDN. One such paper [23] analyzed the SDN architectures: the Path Computation Element (PCE), 4D, and the Secure Architecture for the Networked Enterprise (SANE) using the STRIDE threat analysis methodology. In [8], the paper discussed the state-of-the-art security solutions proposed to secure SDN. The new security advantages that SDN brings and how some of the long-lasting issues in network security can be addressed by exploiting SDN capabilities were discussed in [5].

2.2 Key Technologies in SDN

2.2.1 OpenFlow

OpenFlow is a communication protocol in order to access the forwarding plane network switches or routers in the network. OpenFlow technical architecture is shown in Figure 2. In the OpenFlow 1.0, Header Fields are matching fields. The matching fields consist of: ingress port, Ethernet source, destination and type, Virtual Local Area Network (VLAN) ID, VLAN priority, IP source, destination, protocol and Type of Service (ToS) bits and TCP/UDP source and destination. The counter is used to manage the data packets, which can be used to count the active table entry of each flow table, the duration of each flow, the number of packets received at each port, and the number of packets transmitted per packet. Actions define how to deal with the data packets, such as forwarding to a port, Such as forwarding to a port, modifying data packets or even dropping packets.

The secure channel is an interface between the OpenFlow switch and the controller. The controller manages and controls the switch through this interface, that is to say, the controller insert flow entry through the secure channel into the flow table.

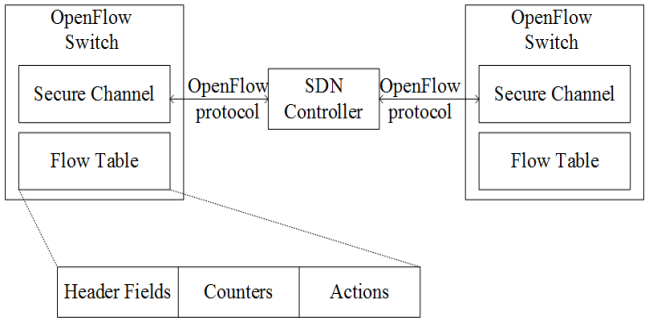

Figure 2. OpenFlow Architecture

In this architecture, when a data packet arrives in the OpenFlow switch through a port, it is compared all the flow entries using the header fields. If this data packet is matched to an existing flow entry in the flow table, the switch updates its counter and executes correspondingly actions. Otherwise, the packet is sent to the controller through the secure channel.

2.2.2 Network Functions Virtualization (NFV)

In the past few years, the design, management and operation of network infrastructure have been developed quickly. The development mainly depends on the innovation of technologies and architectures including SDN and NFV.

NFV was proposed to innovate in the service delivery arena. It uses a standard computing virtualization technology to consolidate in commodity hardware the functions previously performed by specific hardware applications.

NFV aims to offer the ability to reduce the device costs through leveraging hardware with high generality and IT virtualization technologies. SDN and NFV have a lot in common but also exist differences. They can use each other to improve the flexibility of the network and simplify the network [16].

2.2.3 Centralized Control

Logically centralized control can support obtaining the global information of the network resources and have a global allocation and optimization according to the application needs, e.g., traffic engineering and load balancing. At the same time, centralized

control also makes the whole network as an equipment to operate and maintain, so as to improve the convenience of network control.

The key of centralized control is the controller that has a very important role in the network. With the increasing popularity of SDN and the role of the controller, there are a lot of controller software packages. We introduce NOX, a controller based on OpenFlow as an example.

NOX is the first OpenFlow controller. It uses a component architecture that benefits from the advantage of various functional components. The components in NOX can be divided into core components and application components as shown in Figure 3.

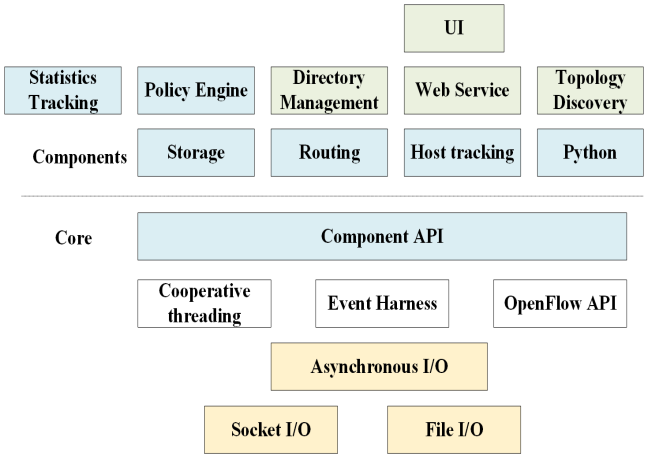

Figure 3. A view of NOX controller

The core components are the underlying interfaces for dealing with the network directly, such as, the message-passing component between a controller and other equipment. The application components include not only the components provided by NOX itself but also the components introducing additional extensions and components developed by users.

2.2.4 Network Programmability

The need for real-time network adaptation means that SDN API must be designed for programs, not humans. Programmability enables automation software that can react to changes and reprogram the network without user involvement in the critical path. Traditional interfaces to network devices are mostly designed for human interaction or narrow management functions. It is, of course, possible to program network devices using existing interfaces, but a well-designed framework makes programming easier and enables a richer ecosystem.

The concept of network programmability lies at the heart of one of the key tenets of SDN. The concept of programmability can exist in, or be a feature of, a number of network devices and software components. This concept is in fact quite different from traditional network management paradigms, where the manager and agent communicated in a relatively loose fashion with considerable lag between operations-including cases where essentially no feedback existed.

In order to realize the paradigm of communication and interaction, tightly coupled, bidirectional programmatic interfaces are needed. These interfaces also need to be readily and rapidly implemented in software so as to encourage their usages and ubiquitous deployment. These interfaces have been commonly referred to as application friendly. These interfaces also need to be developed by communities of developers in order to make them robust, secure, and widely used. This will lead to de facto standardization and ultimately proper standardization. Interface need to provide self-describing capabilities so that applications can easily and dynamically learn and understand the capabilities of a network element without having to be recompiled.

3. SECURITY REQUIREMENTS

As a new technology, SDN also introduces many security threats in spite of many advanced attributes, such as centralized control and network programmability. In this chapter, we go through main security threats specifically relevant to SDN.

3.1 Security Threats in SDN

Forged traffic flow: In a SDN network, when a switch receives a packet, it checks its flow table. If there are corresponding entries in the flow table, it forwards the packet in accordance with the flow table. If there is no corresponding entry in the flow table indicating where to forward the packet, it sends a request to the controller for routing path computation. The controller receives the request from the switch, computes the appropriate routing path, and sends the response to the switch. The controller has a buffer to receive requests from switches. If the buffer is full, new requests will be dropped and will never be served. Due to the centralized nature of SDN, the controller is an ideal target of DoS attack. A DoS attack can be triggered by faulty devices or by malicious users. Malicious users can initialize a large number of flows simultaneously across the networks. A large number of simultaneous packets can overload the controller for computing routing paths, which can cause a large number of packets dropped out of the controller. Forged traffic flows can be used to attack switches and controllers. An attacker can use network elements to launch a DoS attack against OpenFlow switches and controller resources. The SDN should discover the forged traffic flow and minimize the impact of controller DoS attack.

Configuration issues: In SDN, applications programmatically communicate their network requirements and desired network behaviors to the controller via northbound interfaces. In general, the administrator will define security policy that is a function of what connection requests are received. The possibility of multiple OpenFlow applications running on a network controller device introduces a unique policy enforcement challenge: the applications bypass predefined mandatory policies by overwriting flow entries and different application may insert different control policies dynamically, how does the controller guarantee that they are not in conflict with each other. The enforcement of security policies is a serious issue.

Illegal or malicious access: Network resources managed by a controller could be accessible by applications that do not possess the appropriate permission. Switches need guarantees that received commands are from authenticated controllers.

Lack of trusted resources: In order to investigate and establish facts about an incident, we need reliable information from all components and domains of the network. Remediation requires safe and reliable system snapshots to guarantee a fast and correct recovery of network elements to a known working state.

3.2 Requirements of SDN Security

In order to resist the potential threats mentioned above, we summarize the security requirements for SDN.

Availability (Av): Availability ensures survivability of SDN controllers to authorized users. The centralized paradigm of SDN

is a potential vulnerability to the system since attackers may launch DoS attacks against the controller.

Authentication and Authorization (AA): Authentication and authorization are the keys to resist impersonate attacks. SDN applications communicate with the controller and have the ability to program the underlying infrastructure through SDN northbound interface. However, current implementation of these interfaces has several drawbacks. Despite the importance of such critical interfaces, they do not offer security features required to monitor and enforce access control on SDN applications. Without these security features in place, by default, all SDN application have full access to the underlying network enabling the possibility of potential malicious applications.

Confidentiality and Integrity (C/I): Confidentiality and integrity are two basic security properties a secure system should fulfill. In a SDN system, all data, no matter control signaling or user data, transmitted in any security domains should be prevented from malicious eavesdropping, modifying and leaking. Data confidentially and integrity should be ensured. For example, the messages exchanged between the applications and the controller may consist sensitive information, such as the internal topology of networks or sometimes even user data, and therefore needs to be protected. The integrity of the transmitted messages should be ensured as well. This is required in order to protect against a malicious entity that may modify commands issued by a legitimate application, resulting in faulty or harmful messages.

Self-healing (Sh): Self-healing is the ability to provide and maintain an acceptable level of service in terms of faults and challenges to normal operations. In order to realize self-healing in a given communication network, probable errors, faults, challenges and risks have to be identified.

Real-time Anomaly Detection (RAD): Monitoring and measurement of network traffic flows in SDN plays a vital role in management task of a SDN controller for controlling the traffic. The real-time detection of anomalies within the measured traffic volumes is needed to identify possible threats or challenges that are caused by either malicious or legitimate intent.

Configuration issues (C): In order to ensure that the data packets transported between hosts are secure, the security administrator defines security policy. This security policy requires that all the data packets transported from one host to anther host must go through the middlebox such as the firewall for packet scan detection. However, there is a security challenge such as bypasses security policy defined by the security administrator to overwrite flow entry. Therefore, the security policy should be enforced.

4. SOLUTIONS OF SDN SECURITY

In this section, we survey security schemes that have been developed for SDN. We organize the section by reviewing existing related work, discussing whether they can satisfy the aforementioned requirements and analyzing whether they address related security threats based on the SDN architecture. Table I compares comprehensiveness of existing work based on the security requirements.

4.1 Solutions for Overcoming DoS/DDoS Attacks

Braga et al. proposed a lightweight method of DDoS attack detection based on traffic flow features [6]. This method was carried out based on a NOX network. In this network, flow tables were kept with statistics about all active flows. All access to the NOX controller needed to go through an effective way and then was detected by an intelligent attack detection mechanism. The intelligent mechanism employed by this method was based on Self Organizing Maps (SOM), an unsupervised artificial neural network trained with features of the traffic flow. The network traffic was classified into normal and abnormal by SOM, i.e., A potential attack took the statistics of flows as parameters to calculate the SOM. This method could increase the accuracy of detection and reduce false alarms effectively. It enhanced the availability of network resources. However, this method only focused on the detection of DDoS attacks. It did not suggest corresponding solutions once an attack is found. By evaluating this work with the security requirements, we find that this method did not consider or support AA, C/I, Sh, and C.

When it comes to DDoS attacks, One of the significant mitigation survival techniques is load balancing which typically increases maximum load capacity of defended systems. In [26], The authors considered the load-balancing problem that is the opportunities of SDN for survival mitigation under DDoS attacks. Traditional balancing and load balancing between network devices composed of two-level solution. The two levels are the L4 and L7 of OSI model. L4 means transport layer and L7 means application layer. The two-level solution to traffic load balancing consists the L7 load balancing and L4 load balancing. The standard means of splitting the traffic streams between endpoint servers was used in the L7 load balancing and the L4 load balancing to enable splitting the packet between different paths in the network. Using the SDN switch-level flows to redirect traffic is the basis of this algorithm. The traffic based on destination and source IP address information for redirecting. This enables traffic division among different routers in the network regardless of the actual content of the traffic data packets. This solution significantly increases survival time of the defended system under DDoS attacks. It can mitigate the DDoS attacks effectively. However, this algorithm does not consider other levels of OSI model. Obviously, this algorithm does not support AA, C/I, Sh, and C.

Wei et al. proposed a solution named FlowRanger to protect the controller from DoS and DDoS attacks [15]. FlowRanger, a buffer prioritizing solution for controllers to handle routing requests based on their likelihood to be attacking requests, which drives the trust value of the requesting sources. In FlowRanger, a ranking algorithm was proposed to identify malicious users according to their past requests to the controller. Then, the controller utilizes prioritized buffers for requests from different sources. Requests from trusted sources are scheduled in higher priority queues than those with low priority queues. FlowRanger is the first controller that is inserted in the controller and this is the so-called controller-side solution. It can work seamlessly with other solutions (switch-side) to further improve the robustness of SDN against DoS attacks to the controller. Furthermore, in this algorithm, the controller deals with the request with high priority, it does not consider the delay problems. This algorithm does not take AA, C/I, Sh, RAD and C into account.

In [33], a scheduling-based architecture was proposed for SDN controller that leads to effective attack confinement and network protection during DoS attacks. This architecture helps the controller kick out the DoS attacks. The scheme contains most of attack traffic at attack ingress switches so that the SDN networks as a whole can continue normal operation. Only Av and RAD were considered roughly in this work.

4.2 Reliability Solutions

Chandrasekaran et al. presented LogoSDN, a re-design of the controller architecture centering around a set of abstractions to eliminate fate-sharing relationships and make the controller and network resilience to SDN-App failures [22]. This re-design of the controller must satisfy three demands. The first is independence between SDN applications. The second is operations issued by the controlled or SDN-Apps must be bundled in favor of self-healing without sacrificing consistency. The last is an efficient self-healing mechanism that can bring to an initial network state. LogoSDN embodies the described demands by providing an isolation layer called AppVisor and a network-wide transaction system called NetLog, which supports self-healing. This re-design controller did not consider AA, C/I, RAD, and C.

In [18], the network reliability improvement associated with the placement of controller in SDN was focused on. The authors analyzed the impact of controller placement from perspective of interdependent networks on SDN reliability. Then, on an interdependence graph, they defined a reliability metric based on the cascading failure analysis. At last,, to improve reliability, they proposed a partition and selection approach to controller placement. Only Sh was considered roughly in this work.

In [20], The authors argue for the need to consider security and dependability when designing SDN. In order to meet this need, Several threats identified was strongly argued in these networks. At the same time, they have a brief discussion about the mechanism of building a secure and reliable SDN control platform. The purpose of the authors is that this paper will attract attention of SDN community to these issues. Only Sh was considered roughly in this work

4.3 Policy Enforcement

In [14], an architecture to enforce a trusted path for the transfer of sensitive data in the network was proposed. In the SDN network, this architecture allows the request and configuration of trust paths (TP). Trust Path Controller (TPC) which running on the top og the controller is the core of this architecture. A sensitive data transporting through the SDN network and between two applications running in two different is the main questions that the authors considered. In order to ensure that the transmission of sensitive data must conform to a given security policy, a Trusted Path Agent (TPA) is proposed. The function of TPA is to establish TP for the transfer of sensitive data flow. TPA is responsible for sending policy to calculate the path of TPC that meet the policy and use a SDN protocol as OpenFlow to configure this path in SDN. Commonly, TPA has two architectures to send policies to TPC: The Out-of-Band and In-Band. The Out-of-Band means that the policy sending to the TPC through a web service interface and the In-Band means that the policy sending to TPC through a specialized “signaling packet”. C/I was satisfied because of Sensitive data is transferred in a trusted path in the network and the policy was automatically enforced by the TPC in the SDN network. However, Av, AA, RAD and Sh were not supported.

Lara et al. Proposed OpenSec an OpenFlow-based security framework that allows a network security operator to create and implement security policies written in human-readable language [11]. OpenSec converts security policies into a set of rules automatically and inserts them into network devices. The target of OpenSec is to allow network creators create network security policies and implement them. The policies are composed of a set of security policies, a security level and a description of the flow. When detecting malicious traffic, the security can be protected automatically. Security services such as DoS attack detection was provided by the processing units. In a word, OpenSec is a virtual layer between the controller and the user. Under the use of OpenSec, the user can depict a flow according to OpenFlow header fields and regulations which security services must be applied to that flow. The user can also define that when malicious traffic was detected, what actions should OpecSec react. By evaluating this work with the security requirements, we find that OpenSec only focus on the definition of policies. So, this framework can meet C and RAD. The remaining requirements Av, AA, C/I and Sh were not satisfied.

In [3], a comprehensive security architecture designed to provide security services was proposed. In order to ensure that the data packets transported between hosts are secure, the security administrator defined security policy. This security policy required that all the data packets transported from one host to anther host must through the middlebox such as the firewall for packet scan detection. However, there is security challenge such as bypasses a security policy predefined by the security administrator by overwriting flow entry and data eavesdropping by inserting fraudulent flow entries. The authors designed a fine-grained naming scheme for the flow entry. The authors add two new features that are the role of policy creator and security privilege level of role in this architecture. The administrator created a given policy and the role of the creator was defined. The role of the creator was divided into four types: a security administrator, a general administrator, a user, or a guest. Their privileges are reduced in turn and the role with a higher privilege level has been given more right to access the SDN controller. In this paper, the process of reacting to new security attacks automatically and rapidly was described in a detailed way. The first is to detect security attacks and generate corresponding security policies. The second is to send security policies to the controller securely. The third is to convert security policies into new flow entries and insert them into flow table. The forth is to synchronize new flow entries with SDN switches timely and securely. This comprehensive architecture can deal with the challenges efficiently. However, how to integrate their security architecture into current commercial networks is not discussed. By evaluating this work with the security requirements, we found that this method did not consider or support AA, Sh, and Av. However, RAD, C and C/I were well enhanced.

Qazi et al. presented SIMPLE, an SDN-based policy enforcement layer for efficient middle-box-specific “traffic steering” [10]. Software-defined Middlebox Policy Enforcement (SIMPLE) allows network operators specify a logical middlebox routing policy and automatically translates this into forwarding rules that take into account the physical topology, switch capacities, and middlebox resource constraints. SIMPLE was driven by the goal of realizing the benefits of SDN-style control for middlebox-specific traffic steering without mandating any placement or implementation constraints on middleboxes and without changing current SDN standards. The authors analyzed the feasibility of using SDN to simplify middlebox traffic steering. In doing so, they also took a significant step toward addressing industry concerns surrounding the ability of SDN to integrate with exiting infrastructure and support L4-L7 capabilities. The L4 and L7 are two levels of Open System Interconnection (OSI) reference model.

Obviously, Av, C/I, AA and Sh were not supported by this scheme. It only supported C and RAD.

In [32], EnforSDN, a new management approach that exploits SDN principles to decouple the policy resolution layer from the policy enforcement layer in the network service appliances was presented. EnforSDN bring higher throughput and lower latency compared to a baseline SDN network. EnforSDN reduces the overall network appliances load, as well as the forward tables size. Only RAD was mentioned in this paper.

4.4 Authentication Schemes

Banse et al. presented a secure northbound interface to applications [27]. Through this interface, an SDN controller can offer network resources, such as statistics, flow information or topology data to registered SDN applications. The controller service is a web-service comparable to existing controller REST APIs and offers a set of resources representing the SDN networks; An application can register several listeners to receive the topology events, message events and applications events; A trust manager ensures that only authenticated and trusted applications can utilize the interface; Each application that requests access to the interface needs to undergo a registration process; By encrypting the communication between SDN applications and the controller, the confidentiality and integrity of the transmitted messages can be ensured. Obviously, AA and C/I were support by this API. Av, Sh, AD and C were not support by this API.

In [12], An authentication protocol based on SDN network architecture was introduced. This paper applied the ideas of trusted network to SDN. This can increase the credibility of the network architecture. When the controller communicates with a non-trusted third party, the authors introduced a trusted domain authentication protocol for protecting the credibility of the controllers among the whole network architecture. In order to prove the security of this protocol, AVIASP, a security analysis system was used. After analysis, this protocol is effectively for authentication among different trusted domain. However, this paper does not take all the attacks into consideration such as opportunistic collusion. Only AA was considered in this work.

In [29], FlowGuard, a comprehensive framework, to facilitate not only accurate detection but also effective resolution of firewall policy violations in dynamic OpenFlow-based networks was introduced. This paper first introduces forth challenges that are brought by OpenFlow in the process of building the firewall. The FlowGuard includes two components: violation detection and violation resolution. In order to support accurate violation detection and enable network-wide access control, a firewall application needs to not only check violations at the ingress switch of each flow, but also track the flow path and then clearly identify both the original source and final destination of each flow in the network. FlowGuard was implemented in switches and each of the switches should insert this framework. This will cause the waste of network resources. T and Av were considered in this work.

4.5 Other Security Solutions

Betge-Brezetz et al. proposed an approach to deal with the lack of trust in the SDN controller or in their applications [13]. The principle does not relay on a single controller, but on several ones that may come from different providers and run in different execution environments. The network configuration requests coming from these different controllers are then analyzed and if these requests are estimated sufficiently consistent and trustable, then they are actually sent to network equipments. An intermediary layer, called Trust-oriented Controller Proxy (ToCP) and based on a network hypervisor, has then been introduced to make this analysis and decision. The achieved implementation of this layer has allowed showing the feasibility of this approach and providing some first evaluations on induced costs. Further evaluations can be performed in order to deeper analyze the impact on performance and scalability. Security can be further improved by adding for instance a signature on the messages exchanged between the controllers and the ToCP. This approach only takes *Av* into attention.

In [21], a simple analysis for confidentiality and authentication about SDN is proposed. In this paper, the authors made a simple comparison about security between traditional network and SDN. Then, they considered the impact in the process of deploying SDN architecture. They analysis all the component of SDN should meet various demand to ensure the security of the network. At last, the authors discuss protocol security about switch and controller, controller and application, controller and controller. Through this paper, we can have a global view on SDN security. However, It does not discuss the security in detail. C/I and AA were considered in this paper.

Granby et al. Presented software platform named SDN-PANDA [16]. The target of this platform is to provide centralized administration and test the anomaly detection technique. This architecture is made of three modules: network monitoring module, network intrusion detection module and network intrusion reaponse module. Only RAD was support in this paper.

In [28], an adaptive mechanism for dynamically updatig the policies wan presented. The target of this mechanism is to reduce monitoring overhead. This paper improved the existing dynamic rule-updating algorithm. In previous work, rules for expansion and contraction of aggregation policies according to adaptive behavior are defined. This paper represents a work towards reducing the complexity of dynamic algorithm for updating policies of flow counting rules for anomaly detection. Only RAD was considered in this paper.

TABLE I. COMPARISON OF EXISTING WORK BASED ON SDN REQUIREMEN

Ref.	Purpose	Av	C/I	Sh	RAD	C	AA
[6]	Against DoS/DDoS	Y	N	N	Y	N	N
[15]	Against DoS/DDoS	Y	N	N	N	N	N
[26]	Against DoS/DDoS	Y	N	N	Y	N	N
[33]	Against DoS/DDoS	Y	N	N	Y	N	--
[18]	Resilience	N	N	Y	N	N	N
[20]	Resilience	N	N	Y	N	N	N
[22]	Resilience	Y	N	Y	N	N	N
[3]	Policy Enforcement	N	Y	N	Y	Y	N
[10]	Policy	N	N	N	Y	Y	N

	Enforcement						
[11]	Policy Enforcement	N	N	N	Y	Y	N
[14]	Policy Enforcement	N	Y	N	N	Y	N
[32]	Policy Enforcement	N	N	N	Y	N	N
[12]	Authentication	N	N	N	N	N	Y
[27]	Authentication	N	Y	N	N	N	Y
[29]	Authentication	Y	N	N	Y	N	N
[13]	Other Security Solutions	Y	N	N	N	N	N
[21]	Other Security Solutions	N	Y	N	N	N	Y
[16]	Other Security Solutions	N	N	N	Y	N	N
[28]	Other Security Solutions	N	N	N	Y	N	N

5. OPEN RESEARCH ISSUES AND FUTURE RESEARCH DIRECTIONS

5.1 Open Research Issues

According to the above analysis and comparison in Section IV, we find a number of open security issues in SDN.

First, a comprehensive SDN security framework that can resist various attacks has not been investigated in the literature. The existing literature on SDN security has focused only on solving one or several security threats in SDN. As shown in Table I, none of existing work fulfilled all security requirements and resisted all security threats. Hence, building a comprehensive SDN architecture that can deal with all kinds of security threats is needed.

Second, the weakness of the controller has not been overcome. The role of the controller is like the brain to human. Once the controller is hijacked, the whole network is paralyzed. So, lots of attention has been paid to the security of the controller. However, the implementation of the controller is open source, the security solutions associated with controller are normally specific, not universal. Hence, a universal controller that can afford various attacks is needed.

Third, the trustworthiness of controllers should be ensured. The controller sends flow entries to switches and inserts them into flow table. The switches should ensure that the flow entries are from a trusted controller. Trust between applications and controllers is another issue. The controller has to ensure that commands are from authenticated applications that are not hijacked by any attackers. At the same time, applications also need to ensure they do not submit commands to malicious controllers. Hence, building a trust architecture in SDN to ensure trust relationships among applications, controllers and switches is an open issue.

The security research in SDN is still in its infancy. A number of open security issues have not been studied. Many existing solutions need further exploration in order to evaluate their applicability and practicality. Therefore, comprehensive evaluation criteria are requested with regard to the investigation of a security scheme.

The SDN northbound interface is directly linking to business applications. Its design needs to be closely combined with the requirements of business demands and has a variety of features. Different from the sorthbound interface that has already existed an OpenFlow standard, the northbound interface is lack of recognized standards with security protection. The unification of the interface standard brings great convenience to the development and the security of SDN.

5.2 Future Research Trends

All above open issues motivate future research. We further suggest a number of promising research topics about SDN security based on the literature review as below.

First, coordination among controllers is an interesting research direction. The SDN controller introduces single point attack. One solution is not relying on a single controller but on several redundant controllers [13]. The network configuration requests coming from these controllers are compared and, if deemed sufficient consistent and then trustable, they are actually sent to the network. However, the coordination among controllers has not been considered, especially with the concern of efficiency. A scheme that deale with trustworthy and efficient communications and collaboration between controllers is needed.

Second, establishing trust between the application plane and the controller plane is an important issue for the overall trust of the control plane. Network devices should be allowed to associate with controllers dynamically but without incurring in distrust relationships. Establishing trust between controllers and applications can ensure that the commands from applicatione cannot be tampered with and are appropriate given the authorization level of the application and that the application is authenticated.

Finally, middlebox is used to enforce policies in SDN. Some improvement in middlebox management, functionality or flow tracking is suggested by making use of the SDN characteristics. Research should be performed to improve the SDN security by deploying secure middlebox and trustworthy policy fulfillment in SDN.

6. CONCLUSIONS

SDN as a new network architecture framework that decouples the data forwarding and control planes has attracted many attentions. In spite of impressive benefits, SDN is still encountering many security challenges. In this paper, we performed an extensive survey on the security of SDN. We analyzed security threats and suggested security requirements based on the SDN architecture. Our survey has explored that there still a number of open issues that have not yet been seriously investigated. We promoted future research by directing a number of research trends. Significant efforts are still needed in order to secure networks based on SDN.

ACKNOWLEDGMENTS

This work is sponsored by the NSFC (grant U1536202), the 111 project (grant B08038), the PhD grant of the Chinese Educational Ministry (grant JY0300130104), the Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2016ZDJC-06), and Aalto University.

REFERENCE

[1] M. Casado et al., “SANE: A protection architecture for enterprise network,” in *Proc. USENIX Security Symp*., 2006, p. 10.

[2] Kreutz, D., Ramos, F. M. V. and 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, 55-60. DOI = http://dx.doi.org/10.1145/2491185.2491199.

[3] Hu, Z., Wang, M., Yan, X., Yin, Y. and Luo, Z. A comprehensive security architecture for SDN. In *Intelligence in Next Generation Networks* (Paris, France, February 17-19, 2015). ICIN '15. IEEE, Washington, DC, 30-37. DOI = http://dx.doi.org/10.1109/ICIN.2015.7073803.

[4] Tasch, M., Khondoker, R., Marx, R. and Bayarou, K. 2014. Security Analysis of Security Applications for Software Defined Networks. In *Proceedings of the AINTEC 2014 on Asian Internet Engineering Conference* (Chiang Mai, Thailand, November 26-28, 2014). AINTEC '14. ACM, New York, NY, 23-30. DOI = http://dx.doi.org/10.1145/2684793.2684797.

[5] Dabbagh, M., Hamdaoui, B., Guizani, M. and Rayes, A. Software-defined networking security: pros and cons. J. IEEE Communications Magazine. 53, 6 (Jun. 2015), 73-79. DOI = http://dx.doi.org/10.1109/MCOM.2015.7120048.

[6] Braga, R. B., Mota, E. M. and Passito, A. P. 2010. Lightweight DDoS flooding attack detection using NOX/OpenFlow.In *Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks* (Denver, USA, October 10-14, 2010). LCN '10. IEEE, Washington, DC, 408-415. DOI = http://dx.doi.org/10.1109/LCN.2010.5735752.

[7] Lamb, C. C. and Heileman, G. L. 2014. Towards robust trust in software defined networks. In *Globecom Workshops* (Austin, USA, Dec. 8-12, 2014). GC '14. IEEE, Washington, DC, 166-171. DOI = http://dx.doi.org/10.1109/GLOCOMW.2014.7063425.

[8] Akhunzada, A., Ahmed, E., Gani, A., Khan, M. K., Imran, M. and Guizani, S. 2015. Securing software defined networks: taxonomy, requirements, and open issues. *J. IEEE Communications Magazine*. 53, 4 (Apr. 2015), 36-44. DOI = http://dx.doi.org/10.1109/MCOM.2015.7081073.

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

[10] Qazi, Z.A., Tu, C., Chiang, L., Miao, R., Sekar, V. and Yu. M. 2013. SIMPLE-fying middlebox policy enforcement using SDN. In *Proceedings of the SIGCOMM 2013 conference on Computer Communication Review* (New York, USA, October 2013). SIGCOMM'13. ACM, New York, NY, 27-38. DOI = http://dx.doi.org/10.1145/2534169.2486022.

[11] Lara, A. and Ramamurthy, B. 2014. OpenSec: A framework for implementing security policies using OpenFlow, In *Proceedings of the IEEE conference on Global Communications* (Austin, TX, 8-12 Dec. 2014). GLOBECOM'14. IEEE, Washington D.C., 781-786. DOI = http://dx.doi.org/10.1109/GLOCOM.2014.7036903.

[12] Zhou, R., Lai, Y., Liu, Z. and Liu, J. 2015. Study on Authentication Protocol of SDN Trusted Domain. In *Proceedings of 2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems (ISADS)*, (Taichung, 25-27 March 2015). ISADS'14. IEEE, Washington D.C., 281 - 284. DOI = http://dx.doi.org/10.1109/ISADS.2015.29.

[13] Betgé-Brezetz, S., Kamga, G.B. and Tazi, M. 2015. Trust support for SDN controllers and virtualized network applications. In *Proceedings of the IEEE conference on Network Softwarization* (London, UK, 13-17 April 2015). NetSoft'15. IEEE, Washington D.C., 1 - 5. DOI = http://dx.doi.org/10.1109/NETSOFT.2015.7116153.

[14] Betgé-Brezetz, S., Kamga, G.B, Amrani, J. A. El. and Maalmi, O., 2014. SDN-based Trusted Path Control. In *Proceedings of 2014 International Conference and Workshop on the Networ*k of the Future (NOF) (Paris, FR, 3-5 Dec. 2014). NOF'14. IEEE , Washington D.C., 1 - 5. DOI = http://dx.doi.org/10.1109/NOF.2014.7119799.

[15] Wei, L. and Fung, C. 2015. FlowRanger: A request prioritizing algorithm for controller DoS attacks in Software Defined Networks. In *IEEE International Conference on Communications* (London, UK, June 8-12, 2015). ICC '15. IEEE, Washington, DC, 5254-5259.

DOI= http://dx.doi.org/10.1109/ICC.2015.7249158.

[16] Granby, B. R., Askwith, B. and Marnerides, A. K. 2015. SDN-PANDA: Software-Defined Network Platform for ANomaly Detection Applications. In *2015 IEEE 23rd International Conference on Network Protocols* (San Francisco, USA, November 10-13, 2015). ICNP '15. IEEE, Washington, DC, 463-466. DOI = http://dx.doi.org/10.1109/ICNP.2015.58.

[17] Jarschel, M., Zinner, T., Hossfeld, T., Tran-Gia, P., and Kellerer, W. 2014. Interfaces, attributes, and use cases: A compass for SDN. *J. IEEE Communications Magazine.* 52, 6 (June. 2014), 210-217. DOI = http:// dx.doi.org /10.1109/MCOM.2014.6829966.

[18] Guo, M. and Bhattacharya, P. 2013. Controller Placement for Improving Resilience of Software-Defined Networks. In *Proceedings of the ICNDC Conference on Networking and distributed Computing* (Los Angeles, CA, Dec 21 - 24, 2013). ICNDC '13. IEEE, Washington, DC, 23-27. DOI = http:// dx.doi.org /10.1109/MCOM.2014.6829966.

[19] Dotcenko, S., Vladyko. A., and Letenko, I. 2014. A fuzzy logic-based information security management for software-defined networks. In *Proceedings of the ICACT Conference on Adavanced Communication Technology* (Pyeongchang, The Koera, Feb 16 - 19, 2014). ICACT '14. IEEE, Washington, DC, 167-171. DOI = http:// dx.doi.org /10.1109/ICACT.2014.6778942.

[20] Reuther, Diego., Fernando M.V, Ramos., and Verissimo, Paulo. 2013. Towards secure and dependable software-defined networks. In *Proceedings of the Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking* (Hong Kong, China, Aug 16, 2013). HotSDN '13. ACM, New York, NY, 55-60. DOI = http://dx.doi.org/10.1145/2491185.2491199.

[21] Smeliansky, R. L. 2014. SDN for network security. In *Science and Technology Conference* (Moscow, Russia, October 28-29, 2014). MoNeTeC '14. IEEE, Washington, DC, 1-5. DOI = http://dx.doi.org/ 10.1109/MoNeTeC.2014.6995602.

[22] Chandrasekaran, B. and Benson, T. 2014. Tolerating SDN application failures with LegoSDN. In *Proceedings of the third workshop on Hot topics in software defined networking*. HotSDN '14. ACM, New York, NY, 235-236. DOI = http://dx.doi.org/10.1145/2620728.2620781

[23] Klingel, D., Khondoker, R., Marx, R. and Bayarou. K. 2014. Security Analysis of Software Defined Networking Architectures: PCE, 4D and SANE. In *Proceedings of the AINTEC 2014 on Asian Internet Engineering Conference* (Chiang Mai, Thailand, November 26-28, 2014). AINTEC '14. ACM, New York, NY, 15-22. DOI = http://dx.doi.org/10.1145/2684793.2684796.

[24] Mehdi, S. A., Khalid, J. and Khayam, S. A. 2011. Revisiting Traffic Anomaly Detection Using Software Defined Networking. In *Proceedings of the 14th international conference on Recent Advances in Intrusion Detection* (Menlo Park, USA, September 20-21, 2011). RAID '2011, Springer-Verlag, Berlin, Heidelberg, 161-180. DOI = http://dx.doi.org/10.1007/978-3-642-23644-0_9.

[25] Durairajan, R., Sommers, J. and Barford, P. 2014. Controller-agnostic SDN Debugging. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. CoNEXT '14. ACM, New York, NY, 227-234. DOI = http://dx.doi.org/10.1145/2674005.2674993.

[26] Belyaev, M. and Gaivoronski, S. 2014. Towards load balancing in SDN-networks during DDoS-attacks. In *Science and Technology Conference* (Moscow, Russia, Oct. 28-29, 2014). MoNeTeC '14. IEEE, Washington, DC, 1-6. DOI = http://dx.doi.org/10.1109/MoNeTeC.2014.6995578.

[27] Banse, C. and Rangarajan, S. 2015. A secure northbound interface for SDN applications. In *Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA* (Helsinki, Finland, 20-22 Aug. 2015). TRUSTCOM '15. IEEE, Washington D.C. 834-839. DOI = http://doi.acm.org/10.1109/Trustcom.2015.454.

[28] Garg, G. and Garg, R. 2015. Detecting Anomalies Efficiently in SDN Using Adaptive Mechanism. In *Proceedings of 2015 Fifth International Conference on Advanced Computing & Communication Technologies (ACCT)* (Haryana, India, 21-22 Feb. 2015). ACCT'15. IEEE, Washington D.C. 367 - 370. DOI = http://doi.acm.org/10.1109/ACCT.2015.98.

[29] Hu, H., Han, W., Ahn, G., and Zhao, Z. 2014. FLOWGUARD: building robust firewalls for software-defined networks. In *Proceedings of the third workshop on Hot topics in software defined networking* (Chicago, USA, 1 Jan. 2014). HotSDN '14. ACM, New York, NY, 97-102. DOI = http://dx.doi.org/10.1145/2620728.2620749.

[30] Khattak, Z.K., Awais, M. and Iqbal, A. 2014. Performance evaluation of OpenDaylight SDN controller. In *Proceedings of 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)* (Hsinchu, Taiwan, 16-19 Dec. 2014). ICPADS'14. IEEE, Washington D.C. 671 - 676. DOI = http://dx.doi.org/10.1145/2620728.2620749.

[31] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. 2008. NOX: towards an operating system for networks. In *Proceedings of ACM SIGCOMM Computer Communication Review* (July 2008). SIGCOMM'08. ACM, Washington D.C.105-110.DOI = http://dx.doi.org/10.1145/1384609.1384625.

[32] Ben-Itzhak, Y., Barabash, K., Cohen, R., Levin, A. and Raichstein, E. 2015. EnforSDN: Network policies enforcement with SDN. In *Proceedings of 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (Ottawa, ON, 11-15 May 2015). IM'15. IEEE, Washington D.C. 80 - 88. DOI = http://dx.doi.org/10.1109/INM.2015.7140279.

[33] Lim, S., Yang, S., Kim, Y., Yang, S. and Kim, H. 2015. Controller scheduling for continued SDN operation under DDoS attacks. *J. Electronic. Letters*. 51 (Aug. 2015), 1259 - 1261.DOI= http://dx.doi.org/10.1049/el.2015.0334.

[34] Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, Mi., Zolla, J., Hölzle U., Stuart S., and Vahdat A. 2013. B4: experience with a globally-deployed software defined wan. In *Proceedings of the ACM SIGCOMM 2013,* SIGCOMM '13. ACM. New York. 3-14. DOI = http://dx.doi.org/10.1145/2486001.2486019.

[35] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. 2008. OpenFlow: enabling innovation in campus networks. *J. Comput. Commun.* 38 (March. 2008), 69-74. DOI= http://dx.doi.org/10.1145/1355734.1355746.

[36] Rothenberg, C.E., Chua, R. , Bailey, J. , Winter, M. , Corrêa, C. N. A. ; de Lucena, S.C., Salvador, M. R., Nadeau, T. D. 2014. When Open Source Meets Network Control Planes. *J. Com. Society*. 47 (Nov. 2014), 46-54. DOI = http://dx.doi.org/ 10.1109/MC.2014.340.