Mobile Netw Appl (2016) 21:764–776
DOI 10.1007/s11036-016-0676-x

Security in Software-Defined Networking: Threats and Countermeasures

Zhaogang Shu[1] · Jiafu Wan[2] · Di Li[2] · Jiaxiang Lin[1] · Athanasios V. Vasilakos[3] · Muhammad Imran[4]

Published online: 12 January 2016

Abstract In recent years, Software-Defined Networking (SDN) has been a focus of research. As a promising network architecture, SDN will possibly replace traditional networking, as it brings promising opportunities for network management in terms of simplicity, programmability, and elasticity. While many efforts are currently being made to standardize this emerging paradigm, careful attention needs to be also paid to security at this early design stage. This paper focuses on the security aspects of SDN. We begin by discussing characteristics and standards of SDN. On the basis of these, we discuss the security features as a whole and then analyze the security threats and countermeasures in detail from three aspects, based on which part of the SDN paradigm they target, i.e., the data forwarding layer, the control layer and the application layer. Countermeasure techniques that could be used to prevent, mitigate, or recover from some of such attacks are also described, while the threats encountered when developing these defensive mechanisms are highlighted.

Keywords Software-defined networking · SDN · Security · Security countermeasures

✉ Jiafu Wan
jiafu_wan@ieee.org

Zhaogang Shu
zhaogang.shu@gmail.com

Di Li
itdili@scut.edu.cn

Jiaxiang Lin
linjx2015@qq.com

Athanasios V. Vasilakos
vasilako@ath.forthnet.gr

Muhammad Imran
cimran@ksu.edu.sa

[1] Fujian Agriculture and Forestry University, Fuzhou, China

[2] South China University of Technology, Guangzhou, China

[3] Lulea University of Technology, Luleå, Sweden

[4] King Saud University, Riyadh, Saudi Arabia

1 Introduction

With the advent of the era of clouds [1], cloud service providers need to satisfy various network service requirements (such as bandwidth, service quality, safety or reliability) for different users, which requires that the network architecture has high elasticity and flexibility, and that the network resources can be allocated flexibly by way of network function virtualization. However, in traditional network architectures, commonly used closed network equipment (such as routers or switches) have the following drawbacks: a) software and hardware are tightly coupled; b) Network protocols that are too complicated are integrated into the devices; c) Almost all devices are manufacturer-proprietary, which means it is difficult to change their functionality or update them. With the ever increasing network scales, the above characteristics make traditional networks increasingly cumbersome, leading to a situation where cloud service providers cannot customize and optimize network resources effectively according to specific user requirements [2, 3].

The Software Defined Network (SDN) [4] refers to a novel and revolutionary network architecture, which can currently be considered as a best practice for network function virtualization. The basic idea is to strip the complex control logic out from all the network nodes, and form a logical control center to guide the packet forwarding; this change would achieve the goal of controlling all network traffic freely by software programming without changing existing network topology. As one of the most promising technologies, SDN has incomparable advantages over traditional network architectures: a) The decoupling of forwarding and control allows application innovation and equipment upgrades to be independent from each other, which would accelerate the development and deployment of new network applications; b) SDN simplifies the network management model, which allows operators to manage the network conveniently; c) The centralized control logic has a global view of the network, which can provide enough information for the operator to optimize network utilities or improve network performance. Therefore, SDN makes up for the defects of traditional networks, and can meet multi-tenant network requirements in a cloud environment more effectively.

Currently, OpenFlow [5] is the de facto standard of SDN, it was proposed by the research team Cleanslate of Stanford University. The widespread acceptance of OpenFlow by academia and industry makes this SDN standard very successful [6]. In industry, many commercially-oriented SDN-enabled networks have been deployed, such as Microsoft's datacenter network [7] and Google's backbone network [8, 9]. What's more, a lot of SDN-enabled network virtualization software has been significantly developed, such as VMWare NSX [10] and Nuage Networks' VSP [11]. In academia, the top international conference, SIGCOMM, has established a special workshop named hotSDN since August 2012, which calls for papers that report the latest research achievements on SDN. In addition, many well-known universities, such as Stanford and Princeton, have also sprung up research projects related to OpenFlow/SDN, which involves controller design, forwarding performance, routing decisions' optimization, network virtualization applications, programmable wireless networks, energy saving in datacenter networks, etc.

Except for the research topics mentioned above, there is also a research direction that has not attracted much attention, which is the security of SDN [12, 13]. If the security of SDN cannot be ensured, their development will encounter a lot of resistance during the process of replacing traditional network architecture, and even become altogether irrelevant. In the past few years, in order to address security threats of SDN, related working organizations have been founded to study the corresponding security challenges and solutions [14]. At the same time, some solutions against SDN security threats have been proposed, which include controller replication schemes, authentication and authorization mechanisms, schemes to protect controllers against Denial of Service or Distributed Denial of Service (DoS/DDoS) attacks, traffic monitoring and analysis, flow-table overflow attack protection, and others [15].

In this article, we analyze SDN from a security perspective with the objective of shedding light on the new security capabilities and the new security threats that accompany this new architecture. Furthermore, we also survey countermeasures against these threats, through which SDN security can be enhanced. The main contributions of this paper include: a) A comparison of advantages and disadvantages of SDN as opposed to traditional networks regarding security issues from the perspective of overall architecture. b) A dissection of the security threats of SDN from the perspective of different function layers and attack types in detail, and pointing out possible security countermeasures.

The rest of paper is organized as follows: Section II presents an overview of SDN architecture. Section III analyzes security issues of SDN. Section IV discusses SDN security threats and corresponding countermeasures in detail. Section V concludes the paper and points out future research directions.

2 Overview of SDN architecture

SDN is a kind of emerging network architecture which decouples data forwarding from the control logic; currently, these aspects are tightly integrated in traditional network equipment, such as switches and routers. The decoupling of data forwarding and the control logic enables the network control and applications to be programmable [16]. Generally, SDN architecture can be divided into three layers, respectively called the data forwarding layer, the control layer and the application layer from bottom to up, as illustrated in Fig. 1.

2.1 Data forwarding layer

The data forwarding layer consists of many SDN switches, which are physically connected by wired or wireless media. Each switch is a simple device in charge of forwarding network packets and has a forwarding table, named the Flow Table, which contains thousands of rules that are used to formulate forwarding decisions.

Each rule item in the Flow Table is made up of three fields: the action, the counter and a pattern. The pattern field defines the flow pattern, which is basically the set of header field values of the packet. When data packets are received, the switch will search its Flow Table to find a rule that matches the fields.

Once the switch finds such a rule, the counter of the rule increases, and the action corresponding to the particular rule will be performed. Otherwise, the switch will notify the controller to request for help or simply discard the packet. It is worth noting that forwarding rule items are not generated by the switch node itself, but are pushed down by the controller from the control layer.

2.2 Control layer

As the SDN's brain, the control layer manages and controls the entire network. We refer to the network node that implements these functionalities as the SDN controller, and it is

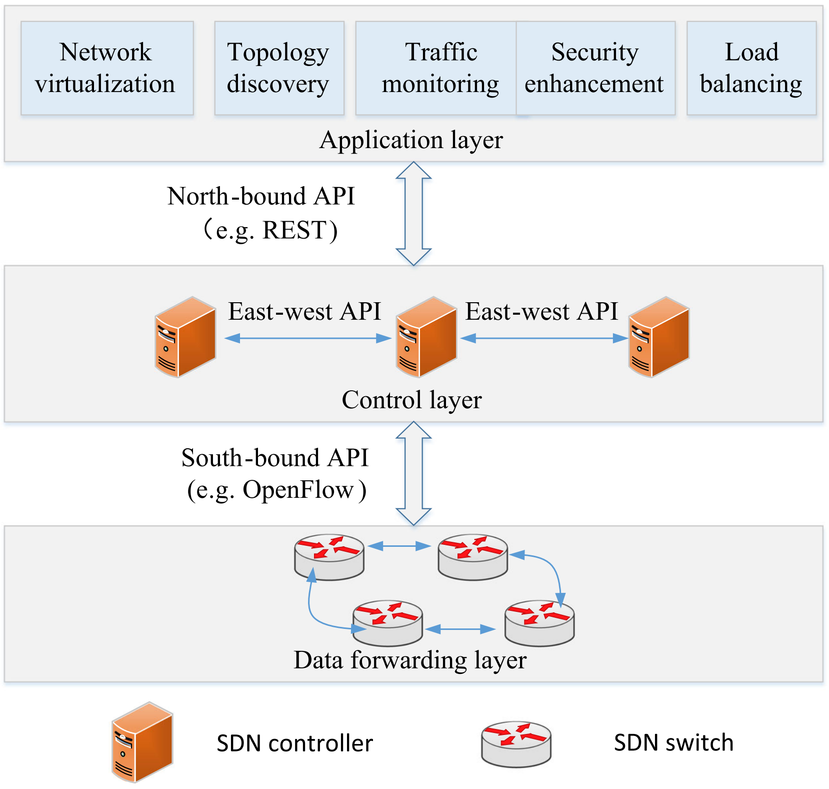

Fig. 1 The architecture of SDN

generally deployed as a separate physical device with specific software. The SDN controller communicates with the switch through a standard south-bound API, e.g., OpenFlow, and has a global view of the entire network topology at the data forwarding layer, i.e., switches and links. Various routing protocols, such as BGP and OSPF, run on the SDN controller so that all the data forwarding taking place in the data layer is based on instructions placed by the controller.

As the de facto standard of SDN, OpenFlow was initially designed with a single controller for simplicity, which constitutes however a potential single point of failure. Therefore, almost all recent SDN architecture implementations, such as Floodlight [17], NOX [18] and OpenDaylight [19], support multiple distributed controllers, which improves the scalability and availability of network resources. In the multi-controller architecture, each individual controller is responsible for controlling only a portion of the switches. In order to maintain the consistency of the network's status and work collaboratively, an individual SDN controller can communicate with other controllers in the network through east–west-bound APIs, as discussed in [20].

2.3 Application layer

The application layer allows network operators to respond rapidly to the various business requirements. Innovative application software has been built to function on top of SDN controllers so that various application requirements are met [21], such as network virtualization [22], topology discovery [23], traffic monitoring [24], security enhancement [25], load balancing [26], and others.

The application layer communicates with the control layer through north-bound APIs, such as the REST API. The control layer provides an abstraction of the network's physical resources for the application layer, which means that network operators can change the data paths of packets using only software programming centrally on the SDN controllers, and not configure all the physical switches in the data path one by one.

3 Security analysis of SDN architecture

Now, we will look into the SDN architecture's characteristics' impact on security. Compared to traditional network architectures, security threats of SDN will be even more concentrated, as opposed to the dispersion seen in the network elements of traditional networks. Therefore, because of its design nature, SDN has security advantages and security defects. Its advantages include:

a) *Effective monitoring of abnormal traffic*. Due to the fact that the SDN controller can perceive the entire network traffic simultaneously, it is easier to notice abnormal behavior in network traffic caused by an attacker.

b) *Timely dealing with vulnerabilities*. Another important advantage can be attributed to the nature of the programmable network environment. Once a new threat has been detected, operators can program new software to analyze and deal with the vulnerability immediately, without spending time to wait for an update of the operating system or application software integrated in the manufacturer-proprietary devices. In addition, the SDN controller can achieve a security policy configuration covering 2–7 layers of the Open Systems Interconnection (OSI) architecture, and provide more granular security control.

On the other hand, the natural security defects of SDN include:

a) *Vulnerable controller*. Most functions, such as network information collection, network configuration, and routing calculation, are concentrated in the SDN controller. The architecture of SDN provides a more concentrated target for, and greatly reduces the difficulty of such attacks. At the same time, the development of cloud computing provides the attacker with very large-scale computing abilities; with the support of cloud computing platforms, attackers can easily implement attacks. If the attackers successfully seize the controller of an SDN, they can cause massive paralysis in network services and affect the whole network covered by the controller.

b) *Risks caused by open programmable interfaces*. Due to their open nature, SDN are more susceptible to security threats. First, it makes the software vulnerabilities of the SDN controller fully exposed to attackers, as the latter will have enough information to formulate an attack strategy. Second, the SDN controller provides a large number of programmable interfaces for the application layer and this level of openness may lead to an abuse of the interface, such as embedding malicious code, such as a virus. Therefore, the open interfaces of SDN controllers need to be carefully evaluated and scrutinized.

c) *More attack points*. As the SDN is divided into three layers, the entities of each layer may be spread across different locations of the network; communication between these entities will be necessary and frequent. Hence, compared to traditional networks, SDN provide more possible attack points for attackers, as shown in Fig. 2. In the figure, we point out six possible attack points in the SDN architecture using red stars, and we will describe them in the order labeling shown.

   - *The SDN switch*. A SDN switch is generally a separate device composed of related hardware and software, which vulnerable to attacks. An example vulnerability is the size limitation of Flow Tables.
   - *The links between SDN switches*. Almost data packets transmitted between SDN switches are not encrypted, and may contain users' sensitive information. These packets can be intercepted by attackers easily, especially when the links between switches are wireless media.
   - *The SDN Controller*. As stated previously, the controller is the most attractive target for attackers. Due to the openness of programmability and complexity of its functionality, the controller's software is inevitably vulnerable, and this can be exploited for malicious attacks.
   - *The links between the controller and the switches*. All forwarding rules are inserted into switches by the controller. The data packets that contain these rules can be tampered with by attacker through eavesdropping on the link between the controller and switch, which will result in a spurious rule insertion or malicious rule modification. Once fraudulent rules are installed in the switch, the data packets will not be forwarded correctly.
   - *The links between controllers*. In a multi-controller environment, the communication between different controllers is necessary for retaining the consistent state of the whole network. The data packets in the links between controllers can be intercepted, which could provide possible clues to attackers for compromising the controllers.
   - *The application software*. The application software is built on the controller directly and is generally located on the same physical device with controller. When the application software invokes the functions of the controllers through the north-bound APIs, malicious code maybe embedded into the controller. Hence, the application software is considered the most convenient attack point for seizing the controllers.

4 Security threats to SDN and corresponding countermeasures

With the advancement of research into SDN, the security issues of SDN attract more and more attention from manufacturers and operators. In this section, we will describe in detail the main security threats and countermeasures that have been presented. According to the above-presented SDN architecture and related security analysis, we divide the threats and corresponding countermeasures into three categories based on which layer of the SDN architecture contains the

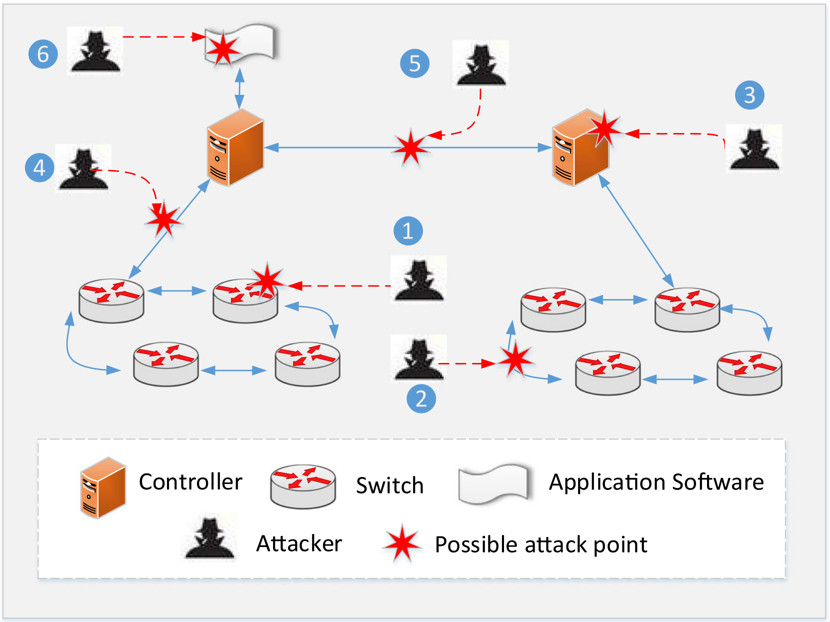

Fig. 2 Possible attack points in the SDN architecture

corresponding attack target, i.e., the forwarding layer, the control layer and the application layer.

4.1 Threats to the data forwarding layer and countermeasures

The data forwarding layer is located at the bottom of the SDN architecture, and contains thousands of switches that are interconnected with each other. These switches are responsible for forwarding packets. If a switch is compromised, the packets that flow through it will not be forwarded correctly. In addition, switches are the direct entry point of network access for end users, and attackers can attack a switch by simply attaching a link to a port of the switch. Therefore, it is very important to recognize security threats and find corresponding countermeasures for SDN switches. We consider the architecture and working principles of SDN switches adhering to the OpenFlow specification, as shown in Fig. 3.

An OpenFlow switch generally contains three function modules, namely the OpenFlow client, the Flow Table and the Flow Buffer. When the switch receives a packet from an input port, it will place this packet in the Flow Buffer, and search the Flow Table to try to find a rule that matches the message fields of this packet, such as a MAC/IP address and a TCP/UDP port. If an appropriate rule is found, the packet will be removed from the Flow Buffer and be forwarded to an output port. Otherwise, the switch will send a Packet_In message through the OpenFlow client to the controller to request instructions. After receiving the new message, the controller would make a routing calculation and insert a new rule into the Flow Table. According to the above process, we can identify three main security threats; they are a man-in-the-middle attack between the switch and the controller, which targets to tamper with rules, a DoS attack to overflow the Flow Table, and a DoS attack to overflow the Flow Buffer.

4.1.1 Man-in-middle attack between switch and controller

(1) Threat description

A man-in-the-middle attack is a classic network intrusion method, the main principle of which is to insert an agent node between the source and the destination node, and is used to intercept communication data and tamper with them without being detected by either communicating sides. Specific attack methods of man-in-the-middle attacks include session hijacking, DNS spoofing, port

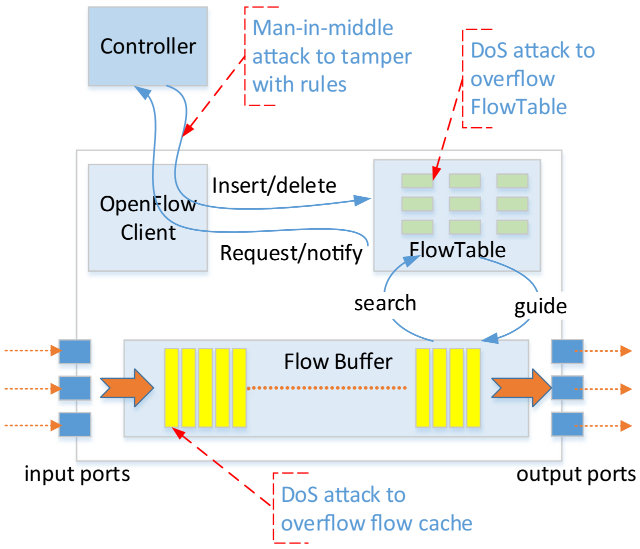
Controller
Man-in-middle attack to tamper with rules
DoS attack to overflow FlowTable
OpenFlow Client
Insert/delete
Request/notify
FlowTable
search
guide
Flow Buffer
input ports
output ports
DoS attack to overflow flow cache

Fig. 3 Working principles and security threats of OpenFlow switches

mirroring, and so on. A man-in-the-middle attack between the controller and switches is an ideal choice for attacking an SDN, as it can be used to intercept and tamper with the forwarding rules issued to the switch in order to gain control of network packet forwarding. After this has been achieved, further attacks can be implemented by attackers, such as black-hole attacks. In addition, we know that the controller and the switches maybe not be directly connected physically, i.e., a packet from a switch to the controller may travel through multiple other switches. Therefore, all switches and the hosts connected to them directly on the communication path are susceptible to be converted to agent nodes in a man-in-the-middle attack.

(2) Countermeasures

In order to guard against man-in-the-middle attack, much research work has been done both in academia and in the industry. The most obvious approach is to create a secure channel between the controller and switches. In OpenFlow specification v1.0, Transport Layer Security (TLS) [27] was used to secure the controller-switch communication. However, the configuration of TLS is very complex, and many vendors do not provide support of TLS in their OpenFlow switches. Thus, later versions of the OpenFlow specification declare that the configuration of TLS is optional. Moreover, TLS cannot provide any TCP-level protection, which means that the network is susceptible to TCP-level attacks. Since TLS is not enforced, our main security challenge in this case is to distinguish between normal and forged flow rules, and to eliminate forged rules before they cause ill effects [28].

Some alternative countermeasures have been presented to this threat. FlowChecker [29] is a configuration validation tool able to recognize internal configuration errors of switches effectively. Specifically, it first creates models of all the interconnected switches, and then executes end-to-end rapid analysis and verification for all switch configurations through a binary decision diagram and model checking technology, by which misconfigurations can be detected. As a software extension module of the NOX controller, FortNOX [30] provides a role-based authorization and authentication security enhancement strategy. Through its novel analysis algorithm, it can detect collisions of various forwarding rules. The algorithm has good robustness and performs its functions correctly even in cases of malicious applications' attack. At the same time, before the applications modify the forwarding rules, FortNOX will verify the legitimacy of the modifications through digital signatures or security constraints. VeriFlow [31] acts as a middle layer between a controller and the switches, and is mainly responsible for the dynamic verification of network variables within the scope of the entire network, especially when a new forwarding rule is inserted. Experiments based on the Mininet, an OpenFlow simulation environment, have been conducted and results show that, by tracking routing data, VeriFlow can finish the detection of a new forwarding rule within a few hundred milliseconds, which is very effective.

Due to the fact that controller connectivity is very important for the proper operation of switches, redundant links or fast link recovery mechanisms are helpful to mitigate the effects of man-in-the-middle attacks between the controller and the switches. The OpenFlow protocol itself has connection stability testing mechanisms, whereby each switch sends activity preserving messages to the controller periodically. If the master controller fails to respond, it can automatically instruct the switch to connect to a backup controller. Controller replication is similar to the mechanism that proposed in [32]. That is to say, if the switch does not receive the response from the controller during a certain period of time, the switch thinks that the controller has failed, and it will quickly establish a connection with another controller, allowing the network to work continuously.

4.1.2 DoS attack to saturate the flow table and flow buffer

(1) Threat description

The reactive rule design of OpenFlow renders the switch vulnerable to Denial of Service (DoS) attacks. Since packets with an unknown destination address will cause a new rule to be inserted in the switch, an attacker can generate large amounts of packets destined to unknown network hosts in a short time, thus quickly filling up a switch's limited Flow Table storage capacity. When the Flow Table is saturated by irregular traffic, legal traffic will not be forwarded correctly, as there will be no more available capacity for inserting new rules.

Except for the Flow Table, another target of DoS attacks is the Flow Buffer. As described above, before packets are forwarded out, they are buffered in the Flow Buffer waiting for the results of the rule search or the insertion of a new rule. Packets in the Flow Buffer will be marked for deletion on a First In First Out (FIFO) basis to release the storage space. As in the case of the Flow Table, the storage capacity of the Flow Buffer is also limited. Attackers can flood large packets belonging to a different flow than that encountered by the switch normally; the switch has to buffer these large packets and this leads to the saturation of the Flow Buffer. When legitimate packets are received, the Flow Buffer will not have enough space to store these packages, and these new packets will have to be dropped.

(2) Countermeasures

We will discuss some examples of related countermeasures that can alleviate this kind of attack. FlowVisor [33] can enable network operators to differentiate network packets according to the header fields of packets. FlowVisor acts as an agent between switches and the controller; it accepts rules from controllers and rewrites them so the resulting rules only affect the portion of the network that a given controller is allowed to control. For example, a controller may be allocated the network segment comprised of all traffic to and from an organization's web servers. This controller might then create a rule to drop all UDP traffic in response to a DoS attack. When FlowVisor receives this rule, it will rewrite it to drop all UDP traffic to and from the web servers, leaving the rest of the network unaffected.

Virtual source Address Validation Edge (VAVE) [34] is a preemptive protection scheme with OpenFlow/NOX architecture aiming to mitigate DoS attacks caused through IP spoofing. A new packet that does not match any rule in the Flow Table will be sent to the controller for source address validation, during which IP spoofing may be detected, in which case the controller creates a rule in the FlowTable to stop the specific flow from that source address. Moreover, address validation in VAVE is very flexible, while the packet process overhead and the required resources are greatly reduced compared to other related works. R. Braga et al. proposed a kind of lightweight defense method against DDoS attack [35], which is based on the characteristics of the traffic flow. This method shows good performance in the analysis of attack detection.

The use of intrusion detection systems could help recognize abnormal traffic flows caused by DoS attack. Such systems could be integrated with related mechanisms for dynamic access control of the switches' behavior, such as rate bounds for control layer requests. Resonance [36] is such a system that can strengthen dynamic access control policy of the controller. The system issues dynamic security policies directly to the forwarding data layer in the SDN architecture based on real-time warnings and packet flow level information.

4.2 Threats to the control layer and countermeasures

In the SDN architecture, the control layer, i.e., the OpenFlow controllers, and its security have a direct impact on the data forwarding layer [37]. If a controller is compromised, the whole network, including a potentially large number of switches, may be affected. This is because if a switch cannot receive forwarding rules from the controller, it will not know how to forward packets. Hence, due to its important role, the controller may become a key target for attackers. The main security threats to and countermeasures of the control layer are described below.

4.2.1 DoS/DDoS attacks on the controller

(1) Threat description

DoS/DDoS attacks attempt to make controller functions unavailable to legitimate users by exhausting computing or memory resources, as shown in Fig. 4.

An attacker could produce enormous flooding traffic in a short time to an SDN-enabled network using their own host or controlling other distributed zombie hosts. This traffic will be mixed together with normal traffic, and it will be difficult to distinguish between the two types. According to the OpenFlow specification, if a switch does not know how to handle a new packet, it will first store this packet in its Flow Buffer and then send a Packet_In message to the controller to request for instructions. Therefore, in the case of a DoS attack, the controller will have to deal with an enormous amount of Packet_In messages generated by the flooding traffic in a short time, which may lead to an exhaustion of resources for processing normal traffic. At the same time, the bandwidth between the controller and the switches may be fully occupied by the attacking traffic and this will seriously reduce the performance of the whole network.

(2) Countermeasures

In order to alleviate the threats of DoS / DDoS on the controller, we can analyze the characteristics of traffic flows stored in the OpenFlow switches to detect this kind of attack. FloodGuard [38] is a SDN-oriented lightweight security framework, which is protocol independent. FloodGuard contains two software modules, the Active Flow Analyzer and Packet Migration, respectively. In order to guarantee the security of a SDN, the Active Flow Analyzer conducts a dynamic analysis based on the real-time running logic of controller, so that it can detect traffic flows caused by DoS attacks. Packet Migration is responsible for buffering the received packets and submitting them to the controller for processing at a limited rate through a rotation scheduling algorithm, which prevents the controller from consuming too much computing resources. When the DoS attack is discovered, the Packet Migration module will redirect a table-miss message to the data forwarding layer. At the same time, the Active Flow Analyzer will monitor the current network flow to determine a variety of sensitive parameters or variables, by which the controller can generate forward flow rules and install them to the switch proactively.

A DDoS is a more potent DoS attack, and its principle is that the attacker can hijack a large number of compromised hosts and control them to produce large-scale

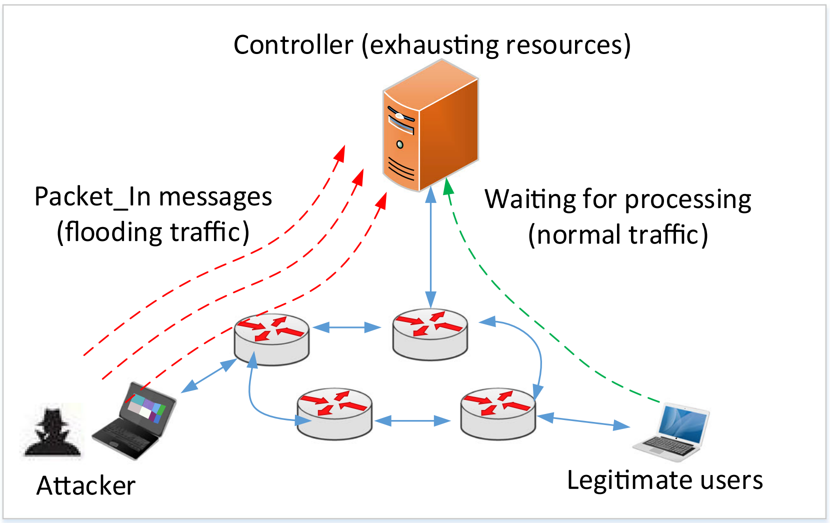
Controller (exhausting resources)
Packet_In messages (flooding traffic)
Waiting for processing (normal traffic)
Attacker
Legitimate users

Fig. 4 DoS / DDoS attack on controller

distributed attack traffic simultaneously to attack a target, such as the controller. In order to cope with DDoS attacks, researchers [39] proposed a DDoS Blocking Application (DBA). The DBA runs on the controller, and distinguishes between normal traffic and attack traffic through a Locator/ID Separation Protocol (LISP) [40]. When the position of a network node changes, the DBA will notify the corresponding change to the controller by locator, which is the clue to discovering the attack. In addition, if the transmission rate of the traffic exceeds a certain value, the controller will consider it as attack traffic and drop these packets directly.

A Content-Oriented Networking Architecture (CONA) [41] is a proxy node located between the client and the content server, and can communicate with the controller. Content request messages from clients are intercepted, analyzed and filtered by CONA, in order to mitigate the harm of DDoS attacks. When the rate of request messages arriving at the content server exceeds a certain value, a DDoS attack is considered to be in progress. The controller will send a message to each relevant CONA agent in order to avoid spreading the attacking traffic and normal messages will be redirected to a new server address.

From the above, we can see that the features of SDN are suitable for defending against DoS attacks, and future solutions for DoS protection may be developed on the application layer of SDN architecture.

4.2.2 Threats on distributed multi-controllers

(1) Threat description

In order to alleviate the risk of having a single point failure in the controller, SDN were originally designed as a single controller architecture, which lacks scalability and reliability. Therefore, the solution of distributed control (controller clusters) has been proposed [42], in which each individual controller instance functions as the master of some switches and different controllers can communicate with each other to collaboratively manage the whole network. However, multiple physical controllers managing the network instead of a single one should be transparent to the data forwarding layer, which means that the controllers need to appear as a single controller for the entire network. In this situation, an application that spans multiple network control domains will need to deal with several security problems, such as authentication, authorization, and privacy issues during network information transmission. In addition, with the distributed collaboration of multiple controllers, the dynamic switch-over of the master controller and the coexistence of multiple controllers in a single network domain can cause configuration conflicts [43]. Therefore, in the multi-controller architecture, an inconsistent configuration is also a hidden security threat.

(2) Countermeasures

DISCO [44] provides control layer functions for distributed heterogeneous networks, and is implemented based on Floodlight [45] with a special protocol, the Advanced Messaging Queuing Protocol [46]. DISCO consists of two modules, an inter-domain control module and an intra-domain control module. The inter-domain control module is responsible for monitoring and managing the priority of data travelling between the domains, so that flow paths with different priorities can be calculated and forwarded. Therefore, the inter-domain control module can dynamically redirect or stop traffic flow to deal with attacks. The intra-domain control module is responsible for managing the communication between controllers, and includes a message transceiver as well as a number of agents. The message transceiver is designed to find neighboring controllers and to provide a control channel regulating the communication between controllers. Agents can exchange the entire network's information using the communication channel provided by the message transceiver module. We can see that the DICSO can effectively deal with security threats faced by distributed controllers.

McNettle [47] is a scalable SDN controller supporting multi-core CPUs, which demonstrates strong extensibility features due to the fact that it allows the addition of all kinds of control algorithms. An operator can also extend McNettle to make it become an advanced programming language for managing the behavior of traffic flow, such as monitoring state changes of each network node and obtaining a global view of the entire network's data flow that can be used to deal with all kinds of malicious attacks effectively. In addition, compared with the NOX controller, McNettle has better performance.

HyperFlow [48] is a scalable event-driven distributed controller platform, in which multiple controllers can operate simultaneously and each controller has the ability to make forwarding decisions locally. In this way, HyperFlow can greatly reduce the time of generating and installing new flow rules, which will improve the performance of the entire control layer.

Load balancing technology [49] can also be used to improve scalability and provide the ability to deal with security threats for SDN controllers. In addition, the total number of controllers and their logical placement will also affect the safety of the controller and its scalability [50].

In [51], the placement of dynamic controllers is discussed and a framework for the dynamic deployment of multiple controllers is proposed. In this framework, the quantity and location of the controllers can be adjusted dynamically according to the network's parameters. The authors of [52] presents a controller optimization framework, which suggests that a single controller should at least meet certain delay requirements, including the communication delay from controller to the switch and between the controllers, so as to effectively deal with related attacks.

4.2.3 Threats from applications

(1) Threat description

Since higher-layer applications can obtain network information only by invoking the API provided by the controller, the latter not only needs to ensure the uninhibited access of legitimate applications, but also prevent malicious or incorrect applications from causing security threats. Applications running on the controller will pose serious security threats to the controller itself. Different applications have different functional requirements, which result in a need to customize a different security policy for each of them. For example, load balancing applications may need to have access to network packet statistics, and intrusion detection applications (IDS) may need to check the header field of packets. Such custom security policies based on different application requirements have not been designed yet.

(2) Countermeasures

Generally, in order to deal with security threats from higher-level applications, related solutions include access authentication of the application, isolation of the resources available to the application, auditing and tracking. The Security-enhanced Floodlight controller (SE-Floodlight) [53], which is based on the original Floodlight controller, features additional security measures; for example, it provides an audit subsystem that tracks all security events so as to detect attacks effectively. SE-Floodlight provides a programmable north-bound API to manage the permissions of applications, which functions as a mediator between applications and controller. SE-Floodlight not only contains an application authentication module that used to verify the integrity of the software modules, but also provides a role-based function authorization module, which can assign different privileges according to the different roles of the applications. FRESCO [54] is a security development framework for OpenFlow applications, which is designed to enable the rapid design and modular composition of software function modules. FRESCO provides many security software modules that implement various security functions, such as attack deflectors, IDS logic, firewalls, scan detectors, and corresponding APIs to invoke these modules. Security applications based on FRESCO can be deployed on any OpenFlow controller to detect and inhibit threats. The authors also demonstrated the utility of FRESCO and its performance.

4.3 Threats to the application layer and countermeasures

In the application layer, attackers can tamper with the network configuration, steal network information, seize network resources and so on through inserting spyware or malware computer programs to the application. In this manner, they can interfere with the normal operation of the control layer and influence the reliability and availability of the network.

Although OpenFlow can deploy security detection algorithms for security applications, these security applications are not mandatory [55]. The variety of applications developed by different independent organizations using different programming languages could produce interoperability inconsistency or security policy conflicts. Some of the security threats to and countermeasures of the application layer are described below.

4.3.1 Illegal access

(1) Threat description

According to the specification of OpenFlow, applications running on the controller are very flexible and extensible and have privileges to access network resources and control network behavior. However, most of these applications are developed by third-party organizations, not controller vendors. Therefore, the lack of a standardized security mechanism for SDN applications causes serious security threats. Security vulnerabilities based on threat vectors in SDN are presented in [56]. The authors opine that it is necessary to design mandatory mechanisms to create a trust relationship between the controller and applications running on it. Although there currently exist various techniques to certify devices in a network, there is no commonly accepted mechanism for verifying network applications.

(2) Countermeasures

PermOF [57] is a fine-grained permission system that can provide privilege control to OpenFlow controllers and applications running on top of it. PermOF summarizes a set of 18 permissions that needed to be enforced by the controller APIs, and also proposes a customized isolation framework that maintains various priorities for applications and isolates the control traffic from the data traffic to achieve comprehensive resource isolation and access control.

NICE [58] is a new solution for enforcing model checking with symbolic execution of event handlers, which can quickly explore the state space of unmodified controller programs written for the popular NOX platform. NICE can also be used as a tool to automate the testing of OpenFlow applications to verify their correctness. Verificare [59] is a tool for modeling distributed systems using formal verification techniques. Authors demonstrated this tool by modeling an OpenFlow network iteratively to verify its correctness and critical properties. VeriCon [60] is a verification system which can confirm the correctness of the controller's programs. VeriCon implements classical FloydHoare-Dijkstra deductive verification through first-order logic and desired network-wide invariants. Experimental results show that VeriCon is able to validate correctness and identify bugs rapidly for in large-scale SDN applications.

Table 1 Security threats and typical countermeasures in SDN architecture

Targeted level	Threats type	Caused by	Typical countermeasures
Data forwarding layer	Man-in-middle attack between switch and controller	• The communication channel is not secure without TLS support	• FlowChecker [29] • ForNOX [30] • VeriFlow [31] • Controller replication [32]
	DoS attack to saturate Flow Table and Flow Buffer	• Limited storage capacity of Flow Table and Flow Buffer • Enormous number of packets in a short time	• FlowVisor [33] • Virtual source Address Validation Edge (VAVE) [34] • Resonance [35]
Control layer	DoS/DDoS attack on the controller	• Attacking traffic can be mixed with normal traffic and is hard to distinguish • Limited computing and storage resources of the controller	• FloodGuard [38] • DDoS Blocking Application [39] • Content-Oriented Networking Architecture (CONA) [40]
	Threats based on distributed multi-controllers	• Distribution of access control • Incorporation difficult for multi-tenant controllers • Inconsistent configuration of multiple controllers	• DISCO [44] • McNettle [47] • HyperFlow [48]
	Threats from applications	• Open programmatic interface • Malicious Applications	• SE-Floodlight [53] • FRESCO [54]
Application layer	Illegal access	• Bypassing of the authentication mechanism • Software vulnerabilities of the controller	• PermOF [57] • NICE [58] • Verificare [59] • VeriCon [60]
	Security rules and configuration conflicts	• Variety for application software • Difference of access control and accountability for various application software	• Flover [61] • Anteater [62] • NetPlumber [63]

4.3.2 Security rules and configuration conflict

(1) Threats description

In order to provide a wide range of network services, the application layer needs to have security applications for accessing the security interfaces of the controller. Along with the complexity of the applications, conflicts may appear between security rules, resulting in a confusion of network services and management complexity.

(2) Countermeasures

Flover [61] is a model checking system which verifies flow policies using assertion sets. Flover is implemented based on NOX using the Yices SMT solver and can provide a formal validation of the functions of an OpenFlow network's security behavior. In addition, Flover supports a batch mode for obtaining responses from a controller quickly.

Anteater [62] proposes a static analysis method for debugging network configuration conflicts, which can offer verification functionality for the data forwarding layer. Generally, it has a short run time, and may detect problems that have occurred rather than issues in real-time for blocking potential disruptions to the network.

NetPlumber [63] is a policy detection tool, which can monitor network consistency caused by an incremental change in the network's state in real-time. Header Space Analysis (HSA) [64] is the theoretical basis for the tool, which is the same authors' previous work and can quickly validate every network status update through incremental calculations based on the dependency sub-graph.

A conceivable scenario is that the firewall is bypassed and switch flow rules are rewritten. In [65], the author applied HSA on the SDN firewall and developed a conflict detection and resolution algorithm, which can create and maintain a dynamic flow graph to detect the flow space and the authorization space of the firewall. Conflicting parts of the flow graph can be avoided by inserting flow-blocking rules or refusing to insert new rules, which will strengthen SDN network security greatly.

4.4 Summarization of security issues

For convenience, Table 1 presents a summary of the security threats that SDN networks are exposed to, along with possible countermeasures that are discussed in this section.

5 Conclusions

In this paper, we concisely reviewed the characteristics and architecture of SDN. We explained how SDN function and analyzed their issues and countermeasures from a security perspective, and gave SDN great security characteristics of the uniqueness and openness. Then, we discussed the state of the art of SDN security and analyzed the security issues from three aspects: the data forwarding layer, the control layer and the application layer. Several preventive and mitigation techniques were also described to address some of those security issues. In the future, network virtualization and middleboxes based on cloud computing will be considered an important application for SDN, which will bring additional security threats. Therefore, the issue of security in these applications is expected to draw increasing amounts of attention.

Acknowledgments This work was supported in part by the Fundamental Research Funds for the Central Universities (No. 2015ZZ079), the Natural Science Foundation of Jiangxi Province, China (No. 20151BAB207024), the Natural Science Foundation of Fujian Province, China (No. 2014J05045), the Natural Science Foundation of Guangdong Province, China (No. 2015A030308002), and the National Natural Science Foundation of China (Nos. 61262013, 61572220, 41401458, 61363011, and 51575194). Imran's work is supported by the Deanship of Scientific Research at King Saud University through Research group No. (RG # 1435-051).

References

1. Chen M, Zhang Y, Li Y, Mao S, Leung V (2015) EMC: emotion-aware mobile cloud computing in 5G. IEEE Netw 29(2):32–38
2. Wan J, Yan H, Suo H, Li F (2011) Advances in cyber-physical systems research. KSII Trans Internet Inf Syst 5(11):1891–1908
3. Suo H, Liu Z, Wan J, Zhou K (2013) Security and privacy in mobile cloud computing. In: Proceedings of the 9th IEEE International Wireless Communications and Mobile Computing Conference, Cagliari, Italy
4. Cisco Inc. (2013) Software-defined networking: why we like it and how we are building on it. White Paper
5. McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Turner J (2008) OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput Commun Rev 38(2):69–74
6. Liu J, Li Y, Chen M, Dong W, Jin D (2015) Software-defined internet of things for smart urban sensing. IEEE Commun Mag 53(9):55–63
7. Hong CY, Kandula S, Mahajan R, Zhang M, Gill V, Nanduri M, Wattenhofer R (2013) Achieving high utilization with software-driven WAN. ACM SIGCOMM Comput Commun Rev 43(4):15–26
8. Google Inc. (2012) Inter-datacenter WAN with centralized TE using SDN and OpenFlow. Open Network Submit
9. Jain S, Kumar A, Mandal S, Ong J, Poutievski L, Singh A, Venkata S, Wanderer J, Zhou J, Zhou M, Zolia J, Hölzle U, Stuart S, Vahdat A (2013) B4: experience with a globally-deployed software defined WAN. In: Proceedings of the ACM SIGCOMM, pp 3–14
10. VMware NSX. [Online] http://www.vmware.com/products/nsx/
11. Nuage Networks VSP. [Online] http://www.nuagenetworks.net/products/virtualized-services-platform/
12. Ahmad I, Namal S, Ylianttila M, Gurtov A (2015) Security in software defined networks: a survey. IEEE Commun Surv Tutorials 17(4):2317–2346

13. Zhang H (2014) A vision for cloud security. Netw Secur 2014(2): 12–15
14. 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, pp 151–152
15. Scott-Hayward S, O'Callaghan G, Sezer S (2013) Sdn security: a survey. In: IEEE SDN Future Networks and Services (SDN4FNS), pp 1–7
16. Pan P, Nadeau T (2011) Software driven networks problem statement. IETF Internet-Draft
17. Floodlight controller documentation for developers [Online]. Available: http://www.projectfloodlight.org/floodlight/
18. Gude N, Koponen T, Pettit J, Pfaff B, Casado M, McKeown N, Shenker S (2008) NOX: towards an operating system for networks. ACM SIGCOMM Comput Commun Rev 38(3):105–110
19. OpenDaylight.[Online]. Available: http://www.opendaylight.org
20. Kreutz D, Ramos FM, Esteves Verissimo P, Esteve Rothenberg C, Azodolmolky S, Uhlig S (2015) Software-defined networking: a comprehensive survey. Proc IEEE 103(1):14–76
21. Lara A, Kolasani A, Ramamurthy B (2014) Network innovation using openflow: a survey. IEEE Commun Surv Tutorials 16(1): 493–512
22. Bernardo DV (2014) Software-defined networking and network function virtualization security architecture. Internet Engineering Task Force. [Online]. Available: https://tools.ietf.org/html/ draft-bernardo-sec-arch- sdnnvfarchitecture-00
23. Yang M, Li Y, Jin D, Zeng L, Wu X, Vasilakos A (2015) Software-defined and virtualized future mobile and wireless networks: a survey. ACM/Springer Mob Netw Appl 20(1):4–18
24. Yuan W, Deng P, Taleb T, Wan J, Bi C (2015) An unlicensed taxi identification model based on big data analysis. IEEE Trans Intell Transp Syst. doi:10.1109/TITS.2015.2498180
25. Jing Q, Vasilakos A, Wan J, Lu J, Qiu D (2014) Security of the internet of things: perspectives and challenges. Wirel Netw 20(8): 2481–2501
26. Namal S, Ahmad I, Gurtov A, Ylianttila M (2013) SDN based inter-technology load balancing leveraged by flow admission control. In: IEEE SDN for Future Networks and Services (SDN4FNS), pp 1–5
27. Dierks T (2008) The transport layer security (TLS) protocol version 1.2 [Online]. Available: http://tools.ietf.org/html/rfc5246
28. Wasserman M, Hartman S (2013) Security analysis of the open networking foundation (ONF) OpenFlow switch specification. Internet Engineering Task Force. [Online]. Available: https://tools.ietf.org/html/ draft-mrw-SDNec-openflow-analysis-02
29. Al-Shaer E, Al-Haj S (2010) FlowChecker: configuration analysis and verification of federated OpenFlow infrastructures. In: Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration, pp 37–44
30. 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, pp 121–126
31. Khurshid A, Zhou W, Caesar M, Godfrey P (2012) Veriflow: verifying network-wide invariants in real time. ACM SIGCOMM Comput Commun Rev 42(4):467–472
32. Fonseca P, Bennesby R, Mota E, Passito A (2012) A replication component for resilient OpenFlow-based networking. In: IEEE Network Operations and Management Symposium (NOMS), pp 933–939
33. Sherwood R, Gibb G, Yap K K, Appenzeller G, Casado M, McKeown N, Parulkar G (2009) Flowvisor: a network virtualization layer. OpenFlow Switch Consortium, Tech. Rep
34. Yao G, Bi J, Xiao P (2011) Source address validation solution with OpenFlow/NOX architecture. In: 19th IEEE International Conference on Network Protocols (ICNP), pp 7–12
35. Braga R, Mota E, Passito A (2010) Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: IEEE 35th Conference on Local Computer Networks (LCN), pp 408–415
36. Nayak A K, Reimers A, Feamster N, Clark R (2009). Resonance: dynamic access control for enterprise networks. In: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking, pp 11–18
37. Shin S, Yegneswaran V, Porras P, Gu G (2013) Avant-guard: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp 413–424
38. Wang H, Xu L, Gu G (2015) FloodGuard: a dos attack prevention extension in software-defined networks. In: 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp 239–250
39. Lim S, Ha J I, Kim H, Kim Y, Yang S (2014) A SDN-oriented DDoS blocking scheme for botnet-based attacks. In: IEEE Sixth International Conference on Ubiquitous and Future Networks (ICUFN), pp 63–68
40. IETF Locator/ID Separation Protocol (LISP) [Online]. Available: http://datatracker.ietf.org/wg/lisp/
41. Suh J, Choi H G, Yoon W, You T, Kwon T, Choi Y (2010) Implementation of a Content-Oriented Networking Architecture (CONA): a focus on DDoS Countermeasure. In: Proceedings of European NetFPGA Developers Workshop
42. Scott-Hayward S (2015) Design and deployment of secure, robust, and resilient SDN Controllers. In: 1st IEEE Conference on Network Softwarization (NetSoft), pp 1–5
43. Li H, Li P, Guo S, Nayak A (2014) Byzantine-resilient secure software-defined networks with multiple controllers in cloud. IEEE Trans Cloud Comput 2(4):436–447
44. Phemius K, Bouet M, Leguay J (2014) Disco: distributed multi-domain sdn controllers. In: IEEE Network Operations and Management Symposium (NOMS), pp 1–4
45. Big Switch Inc. (2012) Developing floodlight modules. floodlight OpenFlow controller Tech. Rep.
46. Advanced message queuing protocol. [Online]. Available: http://www.amqp.org
47. Voellmy A, Wang J (2012) Scalable software defined network controllers. In: Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp 289–290
48. 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. USENIX Association, pp 3–3
49. Liu J et al (2016) Leveraging software-defined networking for security policy enforcement. Inf Sci 327:288–299
50. Heller B, Sherwood R, McKeown N (2012) The controller placement problem. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, ACM, pp 7–12
51. Bari MF, Roy AR, Chowdhury SR, Zhang Q, Zhani MF, Ahmed R, Boutaba R (2013) Dynamic controller provisioning in software defined networks. In: 2013 9th IEEE International Conference on Network and Service Management (CNSM), pp 18–25
52. Hock D, Hartmann M, Gebert S, Jarschel M, Zinner T, Tran-Gia P (2013) Pareto-optimal resilient controller placement in SDN-based core networks. In: 25th IEEE International Conference on Teletraffic Congress (ITC), pp 1–9
53. Security-enhanced floodlight. [Online]. Available: http://www.sdncentral.com/education/toward-secure-sdn-controllayer/2013/10/
54. Shin S, Porras P, Yegneswaran V, Fong M, Gu G, Tyson M (2013) FRESCO: Modular Composable Security Services for Software-Defined Networks. In : Proceedings of Network and Distributed Security Symposium, pp 1-16

55. Shin S, Porras P, Yegneswaran V, Gu G (2013) A framework for integrating security services into software-defined networks. In: Proceedings of the 2013 Open Networking Summit (Research Track poster paper)
56. Kreutz D, Ramos F, 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, pp 55–60
57. 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, pp 171–172
58. Canini M, Venzano D, Peresini P, Kostic D, Rexford J (2012) A NICE way to test OpenFlow applications. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation
59. Skowyra R, Lapets A, Bestavros A, Kfoury A (2013) Verifiably-safe software-defined networks for CPS. In: Proceedings of the 2nd ACM International Conference on High Confidence Networked Systems, pp. 101–110
60. Ball T, Bjmer N, Gember A, Itzhaky S, Karbyshev A, Sagiv M, Valadarsky A (2014) Vericon: towards verifying controller programs in software-defined networks. ACM SIGPLAN Not 49(6):282–293
61. Son S, Shin S, Yegneswaran V, Porras P, Gu G (2013) Model checking invariant security properties in OpenFlow. In: 2013 I.E. International Conference on Communications (ICC), pp 1974–1979
62. Mai H, Khurshid A, Agarwal R, Caesar M, Godfrey P, King S (2011) Debugging the data plane with anteater. ACM SIGCOMM Comput Commun Rev 41(4):290–301
63. Kazemian P, Chan M, Zeng H, Varghese G, McKeown N, Whyte S (2013) Real time network policy checking using header space analysis. In: USENIX Symposium on Networked Systems Design and Implementation, pp 99–111
64. Kazemian P, Varghese G, McKeown N (2012) Header space analysis: static checking for networks. In: USENIX Symposium on Networked Systems Design and Implementation NSDI, pp 113–126
65. Wang J, Wang Y, Hu H, Sun Q, Shi H, Zeng L (2013) Towards a security-enhanced firewall application for openflow networks. In: Cyberspace Safety and Security, Springer International Publishing, pp. 92–103