SECURITY THREATS AND COUNTERMEASURES IN SOFTWARE DEFINED NETWORKING

[1]Karthik Srinivasan, [2]Saravanan Matheswaran, [3]Prabaharan Sengodan
[1]Department of Information Technology, [2,3]Department of Computer Science and Engineering
[1]College of Computing and Informatics, Saudi Electronic University, Riyadh, Saudi Arabia
[2]Aurora's Technological and Research Institute, Hyderabad, India
[3]Jyothishmathi Institute of Technology & Science, Karimnagar, India

Abstract: In recent years, Software-Defined Networking (SDN) has been a focus of research. SDN offers numerous benefits including on-demand provisioning, automated load balancing, streamlined physical infrastructure and the ability to scale network resources as per the need. SDN can manage network traffic through software and the administrator gain a much higher degree of control, which provides the ability to change network rules on the fly. In the near future, SDN can replace traditional networking. At the same time, careful attention needs to be paid to security at this early design stage. This paper mainly focuses on the security aspects of SDN. We begin by discussing the architecture of SDN. According to the design architecture, we discuss the possible security defects in SDN. Finally discuss various threats and its countermeasures based on three-layer architecture, i.e. data forwarding layer, control layer, and application layer. In addition to that various defensive mechanisms are highlighted.

Keywords- Software Defined Networking, SDN, Security, Countermeasures

I. INTRODUCTION

In the traditional networking approach, most of the networking functionalities are implemented in a dedicated appliance and hardware i.e, the switches, routers, etc. Operating and maintaining today's network is an arduous task because of the many complexities and the various policies implemented on it. Today's network is a heterogeneous collection of switches, routers, and systems which uses vendor specific and low-level commands. Implementing new global policies or changing a small set of the device need each device to configure separately which makes the complex and time-consuming task.

As a relatively recent proposal, Software-defined Networking (SDN) has the potential to reduce many of these traditional network problems because of its support to dynamic nature of network, centralized control plane and direct programmability [1] [2]. SDN is an approach to networking that separates the control plane from the forwarding plane to support virtualization. SDN is a new paradigm for network virtualization. Adopting an SDN methodology has a myriad of benefits including flexibility, scalability, redundancy, and performance. Currently, OpenFlow [3] is the de facto standard of SDN. The widespread acceptance of OpenFlow by academia and industry makes this SDN standard very successful [4].

While SDN is promoting many new network applications, security has become a significant concern. SDN and a diverse set of SDN-based security applications will rapidly gain traction in the fight against cybercrime. SDN can make it easier to collect network usage information, which could support improved algorithm design used to detect attacks. The new generation of applications will take advantage of better-informed SDN agents to improve policy enforcement and traffic anomaly detection and mitigation. These applications may be able to block malicious intruders before they enter the critical regions of the network. If the security of SDN cannot be ensured, their development will encounter much resistance during the process of replacing traditional network architecture, and even become altogether irrelevant. The main aim of this paper is to discuss the architecture of SDN, Security defects of SDN and threats with its countermeasures to SDN.

The rest of the paper is organized as follows. To facilitate discussions on SDN security, Section II briefly introduces the architecture of SDN. Section III discusses about security defects of SDN. Section IV focuses on SDN security threats and countermeasures. Section V concludes this paper.

II. SDN ARCHITECTURE

The inflexibility of the traditional network is due to the tight coupling of control plane and data plane. These inflexibilities of traditional network hide the adoption of changing network infrastructure needs. The SDN architecture distributes the tightly coupled control and forwarding logic into different layers. Open Network Foundation has given three-layered reference architecture for the SDN networks [5]. The decoupling of data forwarding and control logic enables network control and applications to be programmable [6]. 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 Figure.1.

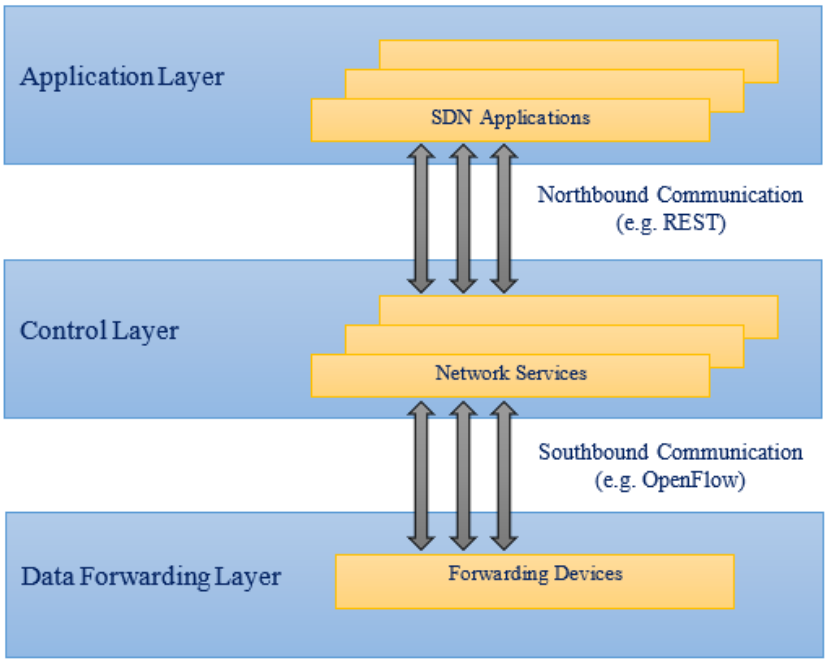

Fig.1 The Architecture of SDN

The Data forwarding layer comprises the network elements, which are merely forwarding devices. This layer consists of many SDN switches, which are physically connected by wired or wireless media. They are exposed to the control layer through the South-bound communication interface.

The Control layer lies in the middle of the architecture and is responsible for translating the requirement of the application thus putting more granular control over the network elements while providing relevant information up to the SDN applications. As the SDN's brain, the control layer manages and controls the entire network. This layer logically centralized to manage all the connected OpenFlow elements in executing out he policies defined on them.

The Application layer has all the SDN applications residing over it. This layer communicates the SDN applications through the north-bound communication interface. The application layer allows network operators to respond rapidly to various business requirements. Innovative application software has been built to function on top of SDN controllers so that various application requirements are met [7], such as network virtualization [8], topology discovery [9], traffic monitoring [10], security enhancement [11], load balancing [12], 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.

III. SDN SECURITY DEFECT ANALYSIS

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 adequate monitoring of abnormal traffic and time dealing with vulnerabilities. So the SDN can effectively notice the abnormal behavior in network traffic created by an attacker and identification of vulnerability exploited by threat in real time.

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

- Vulnerable controller,
- Risks of open programmable interfaces
- More attack points

Vulnerable Controller:

The SDN controller is typically the primary target for attackers because it is the central point for decisions in a network and a central point of failure. Attackers can try to get control of the network by breaking into a controller or pretending to be one. Once a central controller is compromised, an attacker can gain complete control over your network. This would be considered an extreme scenario, but it could be possible as SDN usage continues to grow.

Risks of open programmable interfaces:

Due to their open nature, SDN is 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. Thus, the open interfaces of SDN controllers need to be carefully evaluated and scrutinized.

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 [13]. The more possible attack points of SDN for attackers as shown in Fig.2.

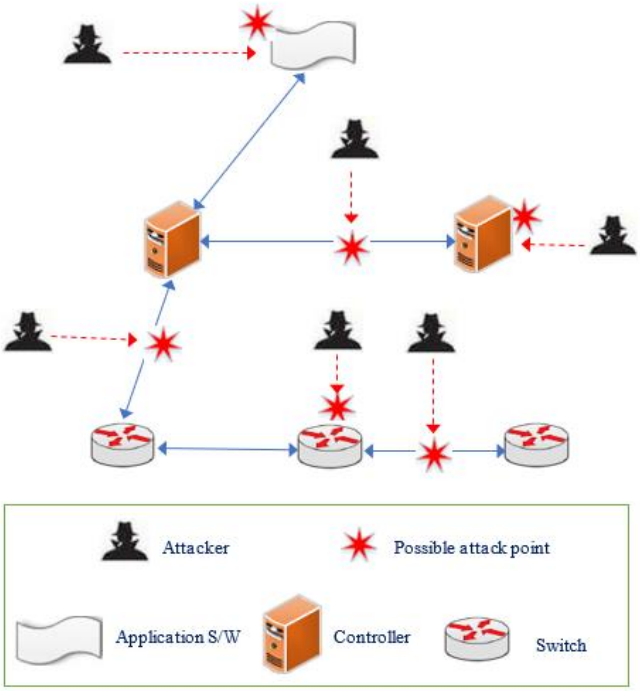

Fig.2 Possible Attack Points in the SDN Architecture

The SDN switch: A SDN switch is generally a separate device composed of related hardware and software, which vulnerable to attacks. An example of 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 the 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 a controller. When the application software invokes the functions of the controllers through the north-bound APIs, malicious code may be embedded into the controller. Hence, the application software is considered the most convenient attack point for seizing the controllers.

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

IV. SDN SECURITY THREATS AND COUNTERMEASURES

In the recent development of SDN, the security issues of SDN gets much more attention from users and manufacturers. In this section, we will discuss the main security threats and countermeasures that have been presented. According to the SDN architecture, threats and its corresponding countermeasures are divided into three categories based on which layer of the SDN architecture contains the corresponding attack target.

A. Threats to the data forwarding layer and countermeasures

The data forwarding layer is placed at the bottom of the SDN architecture and consists of hundreds of switches that are interlinked together. The primary purpose of these switches is to forwarding the packets. The packets will not be forwarded to the next hop if the switch is compromised. Also, a switch is the direct entry device to access the network for the end user. Thus, it is vital to recognize the security threats and find several countermeasures for SDN switches.

Typically, an OpenFlow switch consists of three modules called the OpenFlow client, the Flow Table and the Flow Buffer. When the switch receives a packet from an input port, it will keep the packet in the Flow Buffer and find the corresponding rule for this packet from the Flow Table. If the rule is found, then the packet is forwarded to the respective output port. Otherwise, if the rule is not found, then the request will be sent to the controller through OpenFlow client for a new rule. According to this process, we found the following main security threats; Man-in-the-middle attack (MITM) between the switch and the controller and DoS attack to overflow the Flow Table and the Flow Buffer.

1) Man-in-the-middle attack between the switch and the controller

Threat Description

A MITM attack is a standard network attack that primary purpose is to insert an agent device between two nodes and used to intercept the data communication and tamper with them without knowing either communication parties. Specific attack types of MITM include DNS spoofing, session hijacking, port mirroring and so on. A MITM attack between the switch and the controller is an excellent choice for attacking SDN so that it can be used to intercept and tamper the forwarding rules issued to the switch to obtain control of packet forwarding.

Countermeasures

To protect SDN against a MITM attack, much of the research work proposed both in academia and industry. The most obvious method is to establish a secure channel the switch and the controller. Transport Layer Security [14] was used in OpenFlow specification v1.0, to protect switch-controller communication. However, many vendors do not provide support to TLS because of its complex configuration. So later versions of OpenFlow specifications specify that TLS configuration is optional. FLowCheker [15] is an alternative countermeasure have been proposed to a MITM attack. FLowChecker is a configuration validation tool which can rapidly recognize the internal configuration of interconnected switches and perform analysis of all connected switches. By doing this analysis, any misconfiguration can be detected. FortNOX [16] is another alternative method which supports a role-based authorization and authentication security enhancement strategy. By using this FortNOX algorithm, it can detect the collision of various forwarding rules. VeriFLow [17] acts a middle layer between the switch and the controller for the dynamic verification of new variables within the network.

2) DoS attack to overflow the Flow Table and Flow Buffer

Threat Description

The reactive flow rule of SDN switch leads to vulnerable due to a DoS attack. When the packets arrived with an unknown destination address, the new rule will be inserted to the switch. By taking this as an advantage, an attacker can generate vast amounts of packets destined to unknown network host in a short time. So, within short duration time, the flow table will get overflow. Thus, the legitimate packet will not get a new rule to reach the destination. Except for the overflow of flow table, another target of DoS attack is the flow buffer.

When a new packet arrived at the switch, they are buffered until results of the rule to forward or new to be come. Packets in the buffer are marked as First In First Out (FIFO) basis. The flow buffer is limited in size. Attackers can flood massive number of packets belongs to a different flow. The switch has to buffer all these packets, and soon it becomes full. Due to the overflow of the flow buffer, legitimate packets will not get enough space in the flow buffer, and new packets need to be dropped.

Countermeasures

To alleviate the DoS attack on Flow table, FlowVisor [18] was proposed, and it can enable network operators to differentiate network packets based on the header fields of packets. FlowVisor can behave like an agent between the switches and the controller. It accepts rule from the controller and rewrites them. Thus, the resulting rules only affect the portion of the network given to the controller is allowed to control.

Moreover, another proposal called Virtual source Address Validation Edge (VAVE) [19] is proposed. It 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.

B. Threats to the control layer and countermeasures

The controllers are the brain of the SDN. As per the SDN architecture, the security of the control layer has a direct impact on the data forwarding layer [20]. If a controller is compromised, then the whole network including a potentially huge number of switches will be affected. Because the switch can not receive any forwarding rules from the compromised controller. So the switch does not know where to forward packets. Due to the importance of the control layer, it becomes a key target to the attackers. Distributed DoS (DDoS) attacks and the threat from an application are the primary sources of attack on the control layer.

1) DDoS attacks on the controller

Threat Description

The DDoS attacks attempt to make a considerable number of traffic to the controller which makes the controller unavailable to legitimate users by exhausting the computing and memory resources. An attacker can produce a massive amount of flooding traffic in a short time to an SDN network by compromising a large number of systems called zombies. This enormous traffic mixed with the legitimate packets and it is to difficult to distinguish between the two types. In the case of the DDoS attack, the traffic between switch and controller is entirely occupied that will affect the performance of the whole network.

Countermeasures

To alleviate the DDoS attack on controllers, FloodGuard [21] is proposed. It is a light-weight security framework and protocol independent. It contains two modules, the Active Flow Analyser, and Packet Migration. The Active Flow Analyzer performs a dynamic flow analysis based on the real-time traffic of the controller. So that it can detect the bogus traffic generated by DDoS attack. 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.

Also, to alleviating the DDoS attack on controllers, DDoS Blocking Application (DBA) [22] was proposed. The DBA runs on the controller and detects the attack traffic from the regular traffic by the use of Locator/ID Separation propotol (LISP) [23]. When the position of a network node changes, the DBA will notify the corresponding change to the controller by the locator, which is the clue to discovering the attack.

2) Threats from applications

Threat Description

Higher layer applications can obtain network information by invoking the application given by the controller. Applications running on the controller will get serious security threats to the controller. Different applications have different functional requirements which results need to customize a different security policy for each of them.

Countermeasures

In order to deal with the security theats from higherlevel applications, Security-enhanced Floodlight controller (SEFloodlight) [24] is proposed. SEFLoodlight provides a programmable north-bound API to manage the permissions of applications, which acts as a mediator between application and controller. In addition with application authorization module, the SEFloodlight provides a role-based function authorization module, which can assign various privileges according to the different roles of the applications.

FRESCO [25] 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.

C. Threats to the application layer and countermeasures

The top layer of SDN architecture is the application layer where attackers can tamper the network configuration, steal information of the network and seize the network resources and so on through by inserting malware programs to the application. Like this, attackers can interfere the normal operation of the control layer and affects the performance of the network. Some of the security threats and countermeasures are given below.

1) Illegal access

Threat description

As per the specification of OpenFlow, applications can flexibly run on the controller and have privileges to access network resources. Most of the applications run on the controller are developed by third-party organizations, not venders of the controllers. Therefore, the lack of a standardization causes a serious security threats.

Countermeasures

To alleviate the above mentioned security threat, PermOF [26] is proposed. It is a fine-grained permission system which can provide privilege control to OpenFLow controllers and applications running on top of it. PermOF suggests 18 permissions that proposes a customized framework and isolates the control traffic from the data traffic to achieve resource isolation and accessc control.

Another proposal called NICE [27] proposed to automate the testing of OpenFlow applications to verify their correctness. NICE is a solution for symbolic execution of event handlers, which can quickly explore the various modules of controller programs.

2) Security rules and configuration conflict

Threats description

To provide the wide range of network services, the application layer needs security applications for accessing the security interfaces of the controller. In addition with complexity of applications, conflits may appear between security rules which results confusion of network services and management complexity.

Countermeasures

The flow policies are verified by a checking system model called Flover [28]. It is implemented based on NOX and can provide a formal validation of the functions of an OpenFLow network's security behavior. Also, the Flover uses a batch mode for obtaining responses from the controller.

NetPlumber [29] is a policy detection tool, which can contiuously monitor network consistency caused by an incremental change in the network's state in real time.

V. CONCLUSION

In this paper, we briefly reviewed the architecture and characteristics of SDN. We explained how SDN worked and analyzed the issues and countermeasures from a security perspective, and gave SDN great security characteristics of the uniqueness and openness. Then, we discussed about the security of SDN and analyzed the issues in security 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.

REFERENCES

[1] D. Kreutz, F. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, Software-Defined Networking: A Comprehensive Survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14–76, Jan 2015.

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

[3] Diego Kreutz, M. V.Ramos,Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig: "SDN-A comprehensive survey.", IEEE, 2014

[4] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Turner J (2008) penFlow: enabling innovation in campus networks. ACMSIGCOMM Comput Commun Rev 38(2):69– 74

[5] Open Networking Foundation. <https://www.opennetworking.org/index.php?lang=en/>.

[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] Lara A, Kolasani A, Ramamurthy B (2014) Network innovation using openflow: a survey. IEEE Commun Surv Tutorials 16(1): 493–512

[8] Bernardo DV (2014) Software-defined networking and network function virtualization security architecture. Internet Engineering Task Force.[Online]. Available: https://tools.ietf.org/html/ draftbernardo-sec-arch- sdnnvfarchitecture-00

[9] YangM, 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
[10]. Yuan W, Deng P, Taleb T, Wan J, Bi C (2015) An unlicensed taxi identification model based on big data analysis. IEEE Trans IntellTransp Syst. doi:10.1109/TITS.2015.2498180
[11] 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
[12] Namal S,Ahmad I, Gurtov A, YlianttilaM(2013) SDN based intertechnology load balancing leveraged by flow admission control. In:IEEE SDN for Future Networks and Services (SDN4FNS),pp.1-5
[13] Zhaogang Shu, Jiafu Wan, DI Li, Jiaxiang Lin, Athanasios V. Vasilakos, Mohammed Imran (2016), Security in Software-Defined Networking: Threats and Countermeasures. Springer, Mobile Networks and Applications, DOI 10.1007/s11036-016-0676-x.
[14] Dierks T (2008) The transport layer security (TLS) protocol version 1.2 [Online]. Available: http://tools.ietf.org/html/rfc5246
[15] 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
[16] 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
[17] 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
[18] 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 https://www.networkworld.com/article/3245173/software-defined-networking/secure-your-sdn-controller.html
[19] 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 https://www.networkworld.com/article/2840273/sdn/sdn-security-attack-vectors-and-sdn-hardening.html
[20] 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.
[21] 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
[22] 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
[23] IETF Locator/ID Separation Protocol (LISP) [Online]. Available: http://datatracker.ietf.org/wg/lisp/
[24] Security-enhanced floodlight. [Online]. Available: http://www.sdncentral.com/education/toward-secure-sdn-controllayer/2013/10/
[25] 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.
[26] 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.
[27] Canini M, Venzano D, Peresini P, Kostic D, Rexford J (2012) A NICEway to test OpenFlow applications. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation.
[28] 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.
[29] Kazemian P, ChanM, 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.