A Recent Trends in Software Defined Networking (SDN) Security

Mudit Saxena
Department of Computer Science and Engineering
M.M.M. University of Technology, Gorakhpur-273010, India
Email Id: muditsaxena77@gmail.com

Dr. Rakesh Kumar
Department of Computer Science and Engineering
M.M.M. University of Technology, Gorakhpur-273010, India
Email Id: rkiitr@gmail.com

Abstract– Software defined networking (SDN) has established a new method for creating and administering networks. It is a new architecture which has been designed to enable more agile and cost effective networks. It promotes separation of data plane and control plane. Thereby enabling network switches in the data plane to act as packet forwarding devices. It also uses logical centralized software to control the entire network behavior. SDN introduces new vulnerabilities that are not present in traditional networks such as communication bottlenecks between control plane and data plane. Current research in SDN follows several identifiable trends that are related to the state of deployment of SDN technologies. In this paper, SDN architecture, its salient features along with recent security aspects have been highlighted.

Keywords – Application layer, Control layer, Infrastructure layer, OpenFlow, Open Networking Foundation, SDN, Security.

I. INTRODUCTION

Software Defined Networking has emerged as a promising network architecture nowadays where network control is separated from forwarding and is directly programmable. It separates data plane and control plane of network devices. The data plane represents all of the data that is being forwarded through the network like packets and the hardware that is used to forward it such as switches, routers. The control plane represents all logic and devices that are responsible for making forwarding decisions. Traditional networks combine these two planes on same devices, forcing each device to make its own forwarding decision based on routing protocols. SDN, on the contrary has a centralized control plane that takes into account global view of network to implement its policies. Through separation of data plane and control plane, network control becomes programmable. It abstracts the underlying physical infrastructure and provides a logical centralized view to applications. Control instructions are passed from SDN controller in control plane to data plane through protocols like OpenFlow etc. One can easily infers the behavior of a network by the use of SDN. Abstraction in SDN keeps the complexity of data-paths transparent from operators. Thereby SDN is easy to operate and maintain [2]. Such network provides reusability concept as single high level program can be implemented for multiple data transfers. Due to this feature, SDN is attracting both academic and industry groups a lot.

The most popular implementation of SDN is OpenFlow which implements the SDN architecture by providing a communication protocol that allows a centralized controller to communicate and program the infrastructure layer. OpenFlow controller programs forwarding rules to network devices like switches that specify actions to be performed on packets whose header match against the specific pattern associated with forwarding rule. The controller performs all necessary forwarding decisions, and the switches only apply actions that have been specified by the controller. This design methodology counters several kinds of network attacks, but also opens the risk of new vulnerabilities.

II. SDN ARCHITECTURE

SDN architecture provides a layered architecture comprising separate layers for infrastructure, control and application. These layers are interconnected by different Controller Plane Interfaces viz., CPIs. Control layer sits in between application layer and infrastructure layer. It communicates with both of them through these CPI's. Fig.1 depicts architecture of SDN. Control layer and Infrastructure layer are connected through data-controller plane interface viz., D-CPI whereas Application layer and Control layer are connected by application-controller plane interface viz., A-CPI.

A. Infrastructure Layer

The infrastructure layer comprises of network elements like routers, switches, hubs, bridges etc. It consists of agent, coordinator, master resource database (RDB) and network element (NE) resources. Network element agent executes SDN controller's instructions in infrastructure layer. The Coordinator is connected to OSS (Open Support System). Through coordinator, OSS allocates NE resources to various clients and establishes policy to govern their use. Master RDB is the repository of all resource information known to the network elements. Infrastructure layer provides services to control layer by D-CPI which is also called southbound

978-9-3805-4421-2/16/$31.00 ©2016 IEEE

interface. Through D-CPI, network elements receive the control instructions from SDN controller. According to these instructions, network elements forward packet in the network.

B. Control Layer:

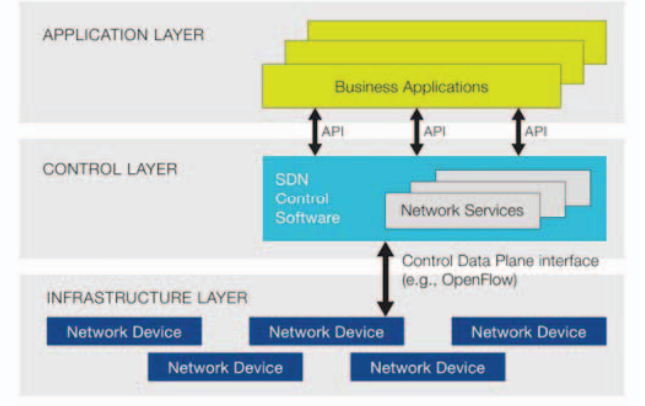
Control Layer forms the core portion of SDN architecture. It consists of SDN controllers which provide centralized control. All programming logic pertaining to packet forwarding, network switching and routing are written dynamically inside SDN Controller. This high level program is transferred to data plane of network devices through D-CPI or southbound interface viz., OpenFlow. In distributed environments, control layer is comprised of multiple SDN Controllers working in synchronization. Control layer agents are used to connect the SDN control logic with Application layer via Programmable API [4]. It presents virtual view of various network resources available in the network to applications and other SDN Controllers. Control layer coordinator performs similar function of management as in infrastructure layer.

C. *Application Layer:*

SDN Application Layer comprised of various applications viz., intrusion detection system (IDS), security system, monitoring services, etc. It also contains general applications requiring access of network devices. The paradigm of SDN architecture, which provides separation of data and control plane from network devices, SDN overcomes different failure of traditional networking, which are:

- Inability to scale with growing demand on data center.
- Due to static nature of current network, it cannot adapt to changing traffic, application and user demands.
- Rate of innovation in networks is slower.

SDN introduced server virtualization that allowed data centers to reach a scale that is unsustainable for traditional networking technologies.



a.

Fig. 1. SDN Architecture

III. FEATURES OF SDN

SDN simplifies the network design and operation. It also simplifies the network devices as the devices no longer need to understand and process protocols but simply accept instructions from SDN controller. Salient features of Distinguishing features of SDN are discussed here:

A. *Logically centralized intelligence*

In SDN model, network control functionality is entirely distributed from forwarding using a standard southbound interface viz., OpenFlow. By centralizing entire network intelligence, decision making is based on a global view of the network. This is opposed to today's prevailing networks. Today's networks are built on an autonomous system view and nodes are not aware about the overall state of the entire network.

B. *Abstraction*

In such network, business applications consuming SDN services are abstracted from various underlying network technologies. The network software is resident in the Control Layer. To ensure portability and future proofing of investments in network services, network devices are abstracted from the control layer.

C. *Programmability*

Software Defined Networks are controlled through software functionality. These softwares are provided by different vendors or the network operators. This kind of programmability makes the management paradigm to be completely replaced by automation. This is influenced by rapid adoption of the cloud. By making available open Application programming interfaces (APIs) for various applications to interact with network, such networks can achieve innovation and differentiation.

D. *Increased network reliability and security*

SDN makes IT to define high level configuration and various policy statements. Which in turn are translated to the infrastructure. Such network architecture completely eliminates requirement of individual configuration of various network devices every time an end point, service, or application is either added, moved. or a policy changes. This reduces the likelihood of network failures because of policy inconsistencies and configuration.

IV. RECENT SDN SECURITY RESEARCH

Recent research into SDN security follows two specific tracks, that of OpenFlow specific research, and general SDN research. As OpenFlow is the most prominent SDN deployment, there is great interest in mitigating security problems in the protocol, as it is used in many production settings. However, other security research in general SDN strategies allow for progress towards new protocols beyond OpenFlow.

A. *General SDN Security*

OpenFlow, despite its popularity, is not the only method of implementing SDNs. There are many limitations of OpenFlow. The work presented in this section lists research that has security applications towards SDNs in general, in contrast to those limited to the OpenFlow specification.

Programming Protocol Independent Packet Processors

The work presented by Bosshart et al. [3] advocates the creation of switch hardware that can support any arbitrary protocol using a parser and a series of match-action processors. This work is motivated by the ability of OpenFlow-enabled hardware to match rules to packets based on different header fields in existing protocols, which allows for rules to be to be enacted on any possible permutations of these fields. However, this work is also motivated by the fact that OpenFlow can only be perform match-actions on protocols that are supported by the specification, such TCP or IP, rather than any arbitrary protocol that a developer desires to support. This work proposes a method to match rules to header fields that are defined by the programmer, which allows the system to support new protocols that have headers that are differently sized then existing protocols. In addition, the system advocates the use of a parser that reads programs that are written in the domain-specific language provided by the authors of the paper, which can be compiled to various hardware platforms. This allows for various different hardware implementations to be supported so long as a compiler can be written for each platform. This ability has several implications for creation of new security measures.

ClickOS

The work presented by Martins, et al. [11] is a method for creating virtualized middleboxes that are capable of operating at line rate. This work is motivated by the Click Modular Router, which allows for the creation of varied packet processing datapaths that can implement various network functions. The researchers note that Click can be used to implement middleboxes in hardware, but the Click software is only implemented inside Linux operating systems, which have a very high system footprint compared to the functionality that is created by a middlebox implemented in Click. The researchers instead use MiniOS, a minimalistic operating system provided by the Xen hypervisor, and run the Click software inside of instances of this operating system. By using MiniOS, the researchers are able to create new virtual machine instances very rapidly, with each potentially implementing different middlebox functionality. This system presents software-defined middleboxes and presents several interesting security opportunities for different SDN technologies.

B. *OpenFlow Security*

As previously stated, the OpenFlow protocol is the deployment of SDN that has received the largest amount of support from the research community and industry. Due to the popularity of the protocol, security issues that were not addressed by the original specification are now priorities of focus, as the protocol is being used in production systems. A list of security vulnerabilities that exist in SDNs, especially the OpenFlow standard many of these vulnerabilities are not unique to SDNs, but these vulnerabilities are oftentimes changes and sometimes exacerbated in SDNs.

Towards Elastic Distributed SDN Controller.

In this work, Dixit et al. [6] present a distributed SDN controller that is compatible with OpenFlow. This work makes significant use of OpenFlow specific mechanisms, especially specific OpenFlow switch management functions. The architecture allows for switches to be migrated between different controllers, and for a pool of available controllers to be resized based on the demand seen in the network. This work is differentiated from others in its use of a distributed data store to store state about the network, and its safety properties that are ensured by the switch migration protocol. This work, along with Kandoo (detailed below), and other distributed controllers, can help to mitigate certain classes of attacks, such as DoS attacks on the control plane. However, the intention of this technology, along with other distributed controllers, is to increase fault-tolerance and scalability of the control-plane, rather than to address security issues.

Kandoo

In this work, Yeganeh and Ganjali [17] present another distributed controller compatible with OpenFlow, which differs from other work in its use of a controller hierarchy. The authors present Kandoo as a system of controllers that establish a hierarchical structure of responsibly, with only the controllers at the top of the hierarchy having responsibly for the entire network. Controllers that exist lower in the hierarchy are responsible only for local decisions, and defer any decisions outside of their scope to the next level, just as the data-plane forwards all decisions to the control plane. The authors only describe a low-level hierarchy with a local and global layer, but they claim that the system can be extended to larger hierarchies if necessary. This system is also useful for defenses against certain types of attack on the control-plane, but its attack surface and response are different than those presented by [6]. It should be noted that security was not one of the primary goals of this work.

DevoFlow

In this work, Curtis et al. [5] detail several shortcomings of the OpenFlow protocol that degrade its performance in high-performance networks, and then present an architecture for a protocol similar to OpenFlow that addresses these shortcomings. The problems noted in Open-Flow by the authors are related to the fact that the control-and data-planes must communicate in order to gather information about flows due to their separation. In smaller-scale networks, this communication is tolerable, but in higher-performance networks, the amount of communication can seriously degrade the performance of the network, which is not just a performance issue, but also a potential security risk. The architecture presented by the authors returns a degree of responsibility to network devices to make local decisions, such as making quick routing decisions, and allows for more efficient collection of statistics in order to reduce communication. In this architecture, the control-plane is only consulted for unknown flows and flows that have been designated for specific traffic engineering goals, such as quality of service (QoS). This is another attempt at reducing the load on the control plane, similar to the previous two works. However, security considerations were made by the authors during the design of the system.

AVANT-GUARD

In this work, Shin et al. [15] present a system that is directly focused on security vulnerabilities that exist in OpenFlow. The system is motivated by two specific vulnerabilities in the protocol, namely the bottle-neck in control-plane and data-plane communication and the slow response rate of the controller to changes in the data-plane. These vulnerabilities are an inherent flaw in the separation of the data- and control-planes, as the controller has to initiate communication to interact with the data-plane, and therefore experiences an unavoidable delay. The researchers propose two mechanisms for the OpenFlow protocol in order to address these issues, which are implemented in the data-plane. The first of these mechanisms is the addition of an ability for network devices to directly respond to TCP connection requests and to filter out illicit connections using SYN cookies and a special TCP hand-shake procedure. In this method, the network switch acts as a proxy for the destination of a TCP request and filters connection requests until it is determined whether these connections will be accepted by both endpoints. Only when it is determined that a connection will be accepted by both endpoints is the control-plane informed of the connection request. The second mechanism is an extension to the flow table rule specification to allow for rules to be triggered by events, such as a threshold number of packets/second. This mechanism also provides the ability for packet payloads and event notifications to be sent to the controller, which is not supported by current OpenFlow specifications.

FRESCO

In this work, Shin et al. [14] present a framework specifically for the development of security applications for OpenFlow. The researchers note the useful features of the protocol, especially the visible that is given to the controller, but also note that creating applications that run on top of controllers entails use of various low-level functions, including the direct writing of flow rules to switches. In order to promote the creation of more OpenFlow-enable security applications, the authors created a framework that provides an abstraction of the OpenFlow protocol that allows for applications to focus on security policies, rather than low-level OpenFlow operations. In order to create this abstraction, the system provides a scripting language and interpreter that allows for applications to compose elements in a fashion similar to Click, and provides an API so that new modules can be written in Python to implement new functionality. This API also allows for non-OpenFlow applications to access the FRESCO abstraction layer. The system also implements a Security Enforcement Kernel that is used to resolve conflicts between different applications based on user privileges, and to ensure that non-security applications do not override or circumvent security applications.

V. CONCLUSION AND FUTURE SCOPE

This paper presents the current research into SDN security. Research has been split between the most popular SDN deployment, namely OpenFlow and general SDN research. As OpenFlow was not originally designed for security and it supports existing protocols, new security research does not seem necessary.

However, the benefits provided by SDN also introduce new security challenges which require further study.

REFERENCES.

[1] S. A. C. Risdianto, E. Mulyana, "Implementation and Analysis of Control and Forwarding Plane for SDN", *IEEE Telecommunication Systems, Services, and Applications (TSSA)* 7th International Conference, pp. 227 – 231, 2012.

[2] Z. Bozakov, P. Papadimitriou, "Towards a Scalable Software-Defined Network Virtualization Platform", *IEEE Network Operations and Management Symposium (NOMS)*, pp. 1- 8, May 2014.

[3] P. Bosshart, D. Daly, and M. Izzard, Programming Protocol-Independent Packet Processors, ACM SIGCOMM Computer Communication Review, Volume 44, Number 3, , pp. 88-95, July 2014.

[4] M. Casado, M. Freedman, and J. Pettit. "Ethane: Taking control of the enterprise," ACM SIGCOMM, 2007.

[5] A. Curtis and J. Mogul, "DevoFlow: scaling flow management for high-performance networks," ACM SIGCOMM, 2011.

[6] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella., "Towards an elastic distributed SDN controller," proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN '13, pp. 7-15, 2013.

[7] N. Feamster, J. Rexford, and E. Zegura, "The Road to SDN," ACM Queue, 11(12), pp. 20-40, Dec. 2013.

[8] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang., "Public Review for A Clean Slate 4D Approach to Network Control and Management," ACM SIGCOMM, 35(5), pp. 41-54, 2005.

[9] E. Kohler, R. Morris, and B. Chen, "The Click modular router," ACM Symposium on Operating Systems Principles, pp. 217-231, December 1999.

[10] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined network," Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking -HotSDN '13, pp. 55-65, 2013.

[11] J. Martins, M. Ahmed, C. Raiciu, and F. Huici., "Enabling fast, dynamic network processing with clickOS.," Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking- HotSDN '13, pp 67-73, 2013.

[12] N. McKeown and T. Anderson, "OpenFlow: enabling innovation in campus networks, " ACM SIGCOMM, 2008.

[13] S. A. Mehdi, J. Khalid, and S. A. Khaya, "Revisiting Traffic Anomaly Detection using Software Defined Networking," Recent Advances in Intrusion Detection (RAID), pp. 1-20, 2011.

[14] S. Shin, P. Porras, V. Yegneswaran, and M. Fong. Fresco, " Modular composable security services for software-defined networks," Network and Distributed Systems Security Symposium, 2(1), 2013.

[15] S. Shin, V. Yegneswaran, P. Porras, and G. Gu., "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks", Proceedings of the 2013 ACM Conference on Computer and Communications Security, 2013.

[16] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden, " A survey of active network research", IEEE Communications Magazine, 35(1), pp. 80-86, Jan. 1997.

[17] S. H. Yeganeh and Y. Ganjali. Kandoo, "A Framework for Efficient and Scalable Offloading of Control Application," Proceedings of the first workshop on Hot topics in software defined networks HotSDN' 12, pp. 19-24, 2012.