International Journal of Computer Networks and Communications Security

VOL. 7, NO. 5, MAY 2019, 83–90
Available online at: www.ijcncs.org
E-ISSN 2308-9830 (Online) / ISSN 2410-0595 (Print)

Network Optimization for Improved Performance and Speed for SDN and Security Analysis of SDN Vulnerabilities

Ayman Haggag

Electronics Technology Department, Faculty of Industrial Education, Helwan University

haggag@techedu.helwan.edu.eg

ABSTRACT

Software defined Network (SDN) has shown a great interest since its emergence. The extreme efficiency in comparison with traditional networks and its agility introduced by separating control and data planes provided many benefits to the market. Yet, there are many challenges accompanied with this agility, leading to many security concerns and threats such as Denial of Service (DoS) and Flooding attacks and many other threats. This paper aims at assessing the effectiveness of this technology via comparisons against traditional networks using standard comparison parameters such as Throughput and Round Trip Time (RTT) and to measuring speed and efficiency of the new underlying technology. Security analysis is carried out to exploit SDN vulnerabilities by implementing successful attacks using primitive SDN Toolkit.

Keywords: *SDN, Control plane, Data plane, Network Topologies, DoS. Flooding Attacks.*

1 INTRODUCTION

In a world its first priority is efficiency, we are always looking for efficient innovation. While Software Defined Networks (SDNs) are well known to be innovative regarding the wide facilities it offers over the traditional networks. SDN already had shown great interests since its arise; although, faces many challenges up to this day; while trying to resolve this tensions, measuring speed and efficiency of the new underlying technology should be regularly.

Software Defined Networking (SDN) has presented itself as the new and most important upcoming network architecture in the communications industry. SDN decouples the network control from the network devices, abstracting it into a single SDN controller, rendering the data plane (or the forwarding plane) no longer part of the network control and any sort of decision making regarding the entire network [1], [2].

SDN allows for new network functions to be introduced to the entire network simply through software-based logic in the network control plane, which, in turn through standard south bound interfaces, implements the software in the data forwarding plane. In SDN the entire network is abstracted in the control plane, giving the network administrator a global view of all the devices in the network [3], mapping the network for the applications that will be running on top of the control plane in the application layer as shown in Figure 1 the three layers/planes of the SDN architecture Figure 1.

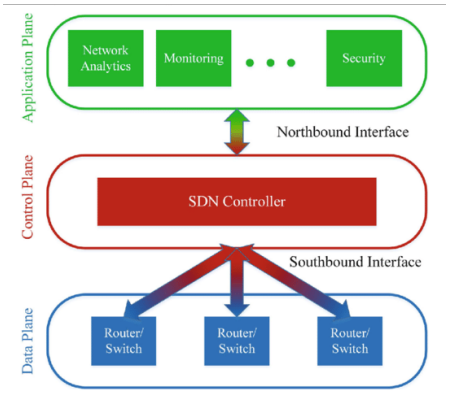


Fig. 1. Three layers/planes of the SDN architecture

This presents the logic of an entire SDN network consisting of:

- The Application Layer: this plane contains all SDN applications with different purposes such as network management, network security or policy implementations.
- The Control Plane: This is the logically centered control point of the network, having a global view of the entire network and providing

hardware abstractions to applications running on top of it in the applications layer [3], [5].

- The Data Plane: it consists of network elements such as routers, virtual switches and physical switches whose main functionality is forwarding data in accordance with the data flow tables and rules provided by the network controller [3], [5].

The centralized nature of an SDN network allows for implementing network-wide security policies and updates simultaneously, entirely removing any risk of policy collision between devices from different vendors. This is due to security applications and systems being mere software-based and only added to the SDN controller, without the need for updating each device like conventional networks, where changing hardware or installing an updated firmware is unavoidable [1], [7]. SDN completely removes the hardships of conventional networks where sets of vendor-specific manually configurable devices spread across the network would compromise their own sub-network with its own set of algorithms for routing, controlling and monitoring the data flow on each device. This combination of numerous sets of different devices from the different vendors present a complexity in combining all the devices seamlessly into the network and prove difficult to maintain a standard set of proprietary protocols, applications or interfaces. As a result, conventional network architectures are unable to provide a global visibility of the network and face difficulties in maintaining network wide policies.

This inability of complete integration prevents healthy and continuous security updates and maintenance of the network, considering the very high costs and time required to manually configure each individual device according to its own vendor specific low-level commands. Moreover, this manual configuration is highly prone to configuration errors or policy conflicts could cause serious security vulnerabilities to the network [8].

On the other hand, on SDN networks, a security application can request flow samples from the network controller, which would return sampled flows from the data plane, the security application would then determine if the data path is to be blocked, rerouted to security middle boxes or even restricting it to specific parts of the network. All of this can be implemented during run-time, on virtue of the centralized logic of the network [9].

However, SDN networks still face their own unique security challenges. Due to the centralized nature of SDN networks, any security compromise of the controller means that consequently the whole network has also been compromised. Furthermore, any breach in the communication between the controller and the data plane can lead to illegitimate access to the network resources. Likewise, the SDN controller allows applications running on top of it access to network resources, the ability to change the behavior of the network and deploying new functionalities. However, an application could prove to be malicious or display abnormal behavior and must be stopped, with security measures applied to contain it, this has also proved to be hard and complex [4].

In this paper, we will discuss in Section 2 the cornerstone of SDNs “the Openflow” protocol. We will compare in Section 3 SDN virus traditional networks in terms of agility and speed. In section 4 we discuss serious threats and challenges facing each individual layer of SDNs. Then, we will simulate an attack in Section 5 exposing SDN vulnerabilities; lacking authentication and the ease of disrupting the network. At the end, we would conclude the paper in Section 6.

2 OPENFLOW

OpenFlow is considered one of the first ever SDN protocols, it is the standard communications interface in SDN environments that allows the SDN controller to directly interact with data plane devices such as routers and switches, both physical and virtual, thus allowing for better configuration to meet the changing business requirements of the network [10].

In a traditional switch both packet forwarding (the data path) and high-level routing (the control path) reside on the same device, however OpenFlow separates the data path from the control path, only leaving the data path on the device and the control path is managed by a separate SDN controller as shown in Fig, 2. The switch and the controller communicate using the OpenFlow protocol [10], [11].

As of today, several established companies such as HP, IBM and google have already either fully utilized, or announced their intention to support, the OpenFlow standard. By early 2012, Google's internal network ran entirely on OpenFlow. OpenFlow is now considered the de facto protocol in SDN communications, thus in this paper we will be discussing the threats that mainly affect all SDN networks, and some specific to OpenFlow enabled SDN networks only.

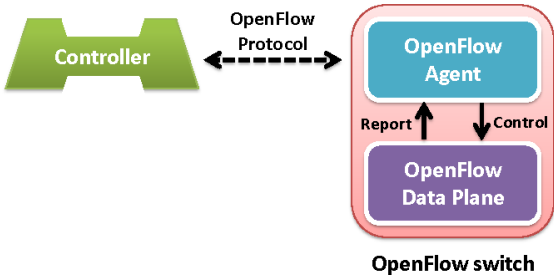


Fig. 2. OpenFlow platform

3 SDN VS TRADITIONAL NETWORKS

One major aspect of recognizing a new technology is determining its efficiency. Therefore, the aim of this paper is to compare SDN with Traditional Networks using various topologies. Measurements of efficiency in this paper are "throughput", which is a measurement of how fast data could be sent across a network and it's measured in Bits per Second (bps) [12], and "Round-trip time" (RTT), which is simply the time required for a packet to travel from a host to another host (t1) and time back to the originating host (t2) [13].

$$RTT = t1 + t2$$

It's worth mentioning that all these experiments were carried out on VMware virtual machine workstation 12 using the virtual image "Mininet" version 2.2.2 that emulates both SDN and Traditional Networks and gives a wide variety of topologies and switches at no cost. Also, the RTT for only the first packet in the 2 technologies is ignored, as it takes more time than other packets to set the network and carry out the required calculations; especially in SDN where the controller takes time in the first packet to populate the switch forwarding table.

Testing the transfer bandwidth and round time trip (RTT) was carried out on two different topologies; Star and Tree. The following tables 1 to 4 represent the transfer rate difference between the different technologies; SDN and Legacy Networks.

Table 1: SDN using star topology

Host 1 to Host 2	Throughput (Gbps)	RTT (ms)
Test 1	30.5	0.060
Test 2	28.5	0.052
Test 3	24.4	0.050
Test 4	27.7	0.048
Test 5	23.8	0.066
Mean	26.98	0.055

Table 2: Traditional Networks using star topology

Host 1 to Host 2	Throughput (Mbps)	RTT (ms)
Test 1	551	0.108
Test 2	508	0.136
Test 3	551	0.112
Test 4	467	0.122
Test 5	526	0.132
Mean	520.6	0.120

Table 3: SDN using tree topology

Host 1 to Host 2	Throughput (Gbps)	RTT (ms)
Test 1	27.6	0.050
Test 2	27.4	0.050
Test 3	23.7	0.046
Test 4	24.2	0.062
Test 5	24.6	0.042
Mean	25.5	0.050

Table 4: Traditional Networks using tree topology

Host 1 to Host 2	Throughput (Mbps)	RTT (ms)
Test 1	554	0.332
Test 2	536	0.106
Test 3	514	0.374
Test 4	587	0.330
Test 5	450	0.104
Mean	528.2	0.249

Considering the results, it is obvious that SDN outperforms traditional networks regarding Throughput. The following chart compares the throughput between the two networks using the different underlying topologies.

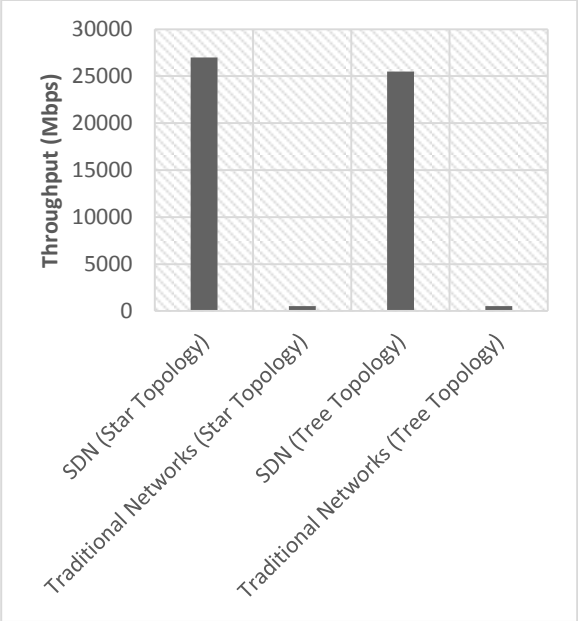


Fig. 3. Comparison of throughput

Considering the Tree Topology, SDN seems to offer much greater throughput than Traditional Networks as shown in Figure 3. The issue was not changed also in the Star topology; concluding that SDN throughput is almost 50 times greater than Traditional Networks regardless the network.

While RTT measures the response time of the network and through a direct measurement of its speed, SDN leads Traditional Networks also here. The following chart shows the mean RTT for both SDN and Traditional Networks for the star and tree topologies.

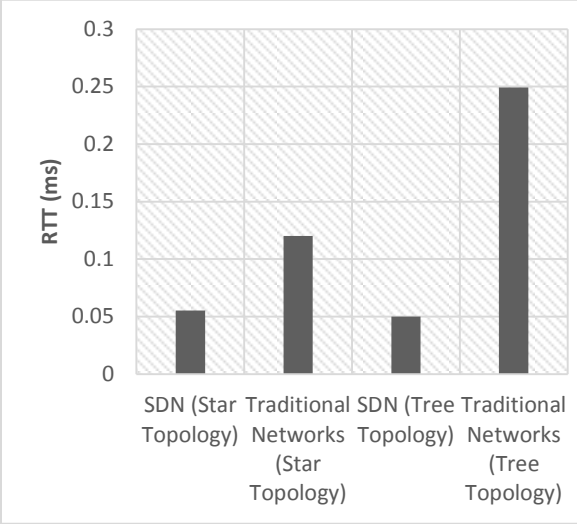


Fig. 4. Comparison of Round Trip Time (RTT)

SDN (Star Topology) RTT is about half the Traditional Networks RTT using the same topology, while the values did not change from star to the tree topology in SDNs, Traditional Networks (Tree Topology) on the other hand seemed to lag the Star topology and seemed to take the time of 5 run time trips of SDN with the same topology as shown in Figure 4.

4 SECURITY AND THREATS TO SDN NETWORKS

While as mentioned before SDN could be separated to three layers. Each particular one faces challenges and threats. For the issue, the problems are classified here regarding the layer which experiences the stated problem in order to address the issue efficiently.

1. Threats to the application layer

It could be said that the ability to control a network using software and the centralization of network intelligence in centralized controllers are the two most prominent features of the SDN architecture, making it innovative and revolutionary, however, these same two features have proved to be the root of serious security challenges to SDN networks. In this section we will be considering security challenges facing the application layer, since most of the network's functions are software applications in this layer of the network.

Even though OpenFlow allows the deployment of security applications, there are no remarkable or leading security applications. Moreover, there are no standard agreed upon development environments or network programming models, as a result the number of vendor or third-party applications developed already were developed using different models, which could result in interoperability problems and security policy collision [14].

In OpenFlow, the applications running on the controller implement most of the functions of the controller, hence, naturally they inherit access privileges to network resources [15], and manipulation of network behavior, all these applications can run without any real security mechanisms from protecting the network from any malicious activity. Therefore, authentication of applications in centralized SDN networks has until now proved to be a challenge. This could be a direct result of the lack of any reliable mechanism to establish a trust relationship between the controller and applications in the SDN environment.

Furthermore, access control in SDN networks is not implemented very carefully. For instance, a malicious application that does not have access to network resources can call an instantiation of another application that has the required access privileges, and thus bypass any security measures quite easily, creating a breach in the network.

2. Threats to the control plane

In a typical SDN architecture the SDN controller is a centralized omnipotent like entity that carries all the decision-making responsibilities of the network, therefore making it a very attractive target for malicious activities due to its pivotal role. On account of the controller's place in the network it can be targeted from both applications running on top of it or through the data plane.

The controller's ability to authenticate applications and authorize any usage of network resources with accurate tracking, auditing and isolation, are what determines the level of controller security from malicious applications [16]. Hence, it is necessary to categorize applications according to their level of security implications before allowing any access to the

network; varying applications with different functional requirements must merit different levels of security clearances, for example, user applications or third-party applications must logically carry a lower security clearance from the controller. Therefore, it is obvious that there is a requirement for each application to have its very own unique customized set of privileges.

Simultaneously, since most of the network's complexity is removed from the forwarding devices and delegated to the controller in accordance with the main concepts of the SDN architecture, there are problems that will arise as the network is scaled. If a controller is asked to determine and install flow rules for each new flow in the incoming data path, the controller's limited capability to handle a huge number of requests in a short interval will cause a bottleneck in the network [17]. Thus, the number of forwarding devices that the controller is responsible for will greatly affect its performance; the larger the number of devices the higher risk of a situation where the controller will require a longer the delay in the period of time to send flow routes to the devices, which can greatly affect the overall performance of the network, and could pose an even more serious risk of the controller not being able to handle all the requests and becoming a single point of failure for the whole network. To counter this risk, it was suggested using more than a single controller, however, in case a single controller was to fail its devices will be allocated to the other controllers, which leaves us at a higher risk that another one will fail next due to the added load, and so on, thus, we will have a series of failing controllers without any difference at all [18].

Similarly, in a DoS or a distributed DoS attack, an attacker could easily trigger control plane saturation by flooding the controller with fraudulent requests that will cause it to crash. Another DoS attack is where the controller will be bombarded with IP packets with random headers that will eventually render the controller unresponsive or broken, and in case more than one controller is used we will have another series of controller breakdowns as the flow of malicious data will be transferred to target the next controller [19].

Another challenge that will arise in the case of an SDN network with multiple controllers and consequently the division of the network into smaller SDN sub-networks, in such situation an application will be implemented in the application layer over the entire network which has a global view of all the sub-domains and it is its job to optimize traffic across several different domains for the other applications. Such application that has access and control over multiple domains might pause an extremely dangerous security threat in case of a breach, in this case, careful and thorough authorization and authentication of this application will be necessary to allow access to privileges that are only needed, to mitigate the risk of any breach either by an attacker or another malicious application in the application layer using this application to gain unauthorized access to network resources and sensitive information [20].

Furthermore, in SDN networks with multiple controllers, any inconsistency in controller configuration could cause delays in network output or cause applications such as firewalls to behave incorrectly.

3. Threats to the data plane

Each network device in the data plane uses its own flow table stored in its hardware to determine packet routing destinations, the size of the flow table however, is limited by its own physical memory hardware, thus in case a switch does not find the specific flow rule for a packet it received it will query the controller for the identity of the destination and route this packet to the destination it receives, but, since all decision making capabilities were removed from individual switches and assigned to the controller, if the network were to be compromised and an attacker were to send the switch a false destination, the switch has no ability to differentiate between genuine and fraudulent flow rules it receives. Additionally, since a switch has to buffer flows it cannot route till it receives further instructions from the controller, in the case of a saturation attack, if the controller was to breakdown this would practically render all forwarding devices – and thus the whole network – offline, since they cannot forward packets without instructions from the controller [4].

In the original OpenFlow design, Transport Layer Security (TLS) and datagram Transport layer Security (DTLS) are defined for south bound communications – communication between the controller and the devices in the data plane [21], [22]. However, in recent OpenFlow versions the use of TLS and DTLS has become optional, and on account of the complexity of configuration of TLS, many vendors have opted to not include TLS altogether on their OpenFlow switches, making the communication channels between the separate planes even more insecure and vulnerable to attacks [23]. Table 5 summarizes the major security threats in each of the SDN planes.

Table 5: Table 5: Summary of major security threats in each of the SDN planes.

SDN Plane	Threat	Description
Data	Flooding attacks	Switches' flow tables only hold limited number of flow rules
	Man-in-the middle attack	As TLS is an option rather than a standard.
	Controller hijacking or failure	Data plane is mainly dependent on the controller which adds another risk to the data plane
Control	DoS attacks	No considerable authentication
	Unauthorized controller access	No compelling user access rights
	Scalability and availability	Increasing size and shear of network would introduce challenges
Application	Lack of authentication and authorization	No means of authentication are carried out on applications
	Fraudulent flow rules insertion	Connected malicious applications may insert false flow rules in the flow tables
	Lack of access control	It's difficult to implement access control

5 DENIAL OF SERVICE ATTACH SCHEME

As in [24], R. Kandoi and M. Antikainen had introduced how the SDN network could be flooded by a hacker to disrupt the switch from sending further traffic; made by simply sending dynamic packet headers that may change the source million times a minute resulting in overflow in the switch flow tables which are populated by the controller; getting the switch out of service as it won't be able to reply to the flooding host due to the high traffic aimed to be handle by the controller and flow rate tables which are over populated with traffic.

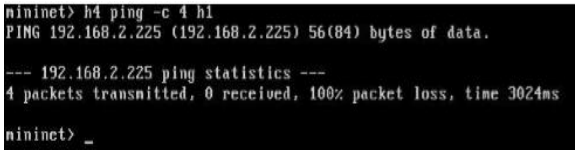
However in this paper, we will clarify a new threat of disrupting the network and gaining access by just using the primitive SDN-toolkit found at [25]. The first step in implementing the attack would be identifying an Openflow based network. The attacker would give some suspected networks to "of-check" [25], which identifies the Openflow-based services by exchanging Hellos messages. If the "of-check" receives a feature request then it would be a controller, otherwise it's a legacy switch.

After determining the target, the attacker would apply "of-enum" [25] to enumerate the Openflow-based network endpoints and identify the specific version which the attacker would be dealing with. It's to mention also that this won't take any effort as "of-enum" is compatible with all Openflow versions.

After identifying and determining the target, the attacker would perform reconnaissance; using "of-map" [25] and connect to the controller as a local host. And it is here to mention the vulnerable drawback of SDN controllers as for instance the two familiar SDN controllers; Floodlight has no authentication at all and Opendaylight has authentication but it is very basic. This paper targets Floodlight controllers. However, Opendaylight controllers' authentication doesn't guarantee security though and could be bypassed also using indirect methods such as exploiting the controller services; granting control on the network.

After connecting to the controller as a local host, the attacker would request Access Control lists (ACLs); identify any targets and sensors. And thereby, the attacker would try to hide any sensor that could detect or deny them and isolate administrator from the network assets and acquiring those assets instead. All these would be implemented using "of-access" [25] which sets access rights and could hide any traffic from any sensor on the network.

And after completing the attack and trying to ping the admin; the result would show that the attack had succeeded and the admin is totally isolated from the network Figure 5.



6 CONCLUSIONS

This paper has introduced the basic functionality and methods of SDN networks in general, and OpenFlow enabled SDN networks in particular, and delved deeper into the security threats that could face each plane of any given SDN network, detailing all types of attacks that could compromise the system, or even logic flaws in the implementation of current SDN protocols and systems that could result in potential breaches or attacks.

Although SDN is superior to traditional networks in terms of agility and speed, it is necessary that all security risks and threats associated with this architecture be carefully analyzed, and solved. We recommend the obligatory usage of TLS, authentication and authorization techniques and application access rights implementation. However, these proposed solutions are always acted upon as suggestions more than critical steps to eliminate the aforementioned security issues.

7 REFERENCES

[1] J. Akrele; Performance Analysis of Openflow-Enabled Network Topologies; www.researchgate.net ; DOI: 10.13140/RG.2.2.33523.84008; Oct, 2018

[2] D. Kreutz, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, & S. Uhlig; „Software-Defined Networking: A Comprehensive Survey,“ arXiv: 1406.0440v3; Oct, 2018G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. (references)

[3] A. Nehra, M. Tripathi, & M. Gaur; “Global View in SDN: existing implementation, vulnerabilities, & threats,” SIN’17, PP: 303-306; 2017

[4] I. Ahmad, S. Namal, M. Ylianttila, & A. Gurtov; “Security in Software Defined Networks: A Survey,” IEEE Communications Surveys & Tutorials Fourth quarter 2015, VOL. 17, pp. 2317-2346; DOI: 10.1109/COMST.2015.2474118; Aug, 2015

[5] W. Xia, Y. Wen, C. Foh, D. Niyato, & H. Xie; “A Survey On Software-Defined Networking,” IEEE Communications Surveys & Tutorials First quarter 2015, VOL. 17, pp. 27-51; DOI: 10.1109/COMST.2014.2330903; June, 2014

[6] B. Nunes, M. Mendonca, X. Nguyen, K. Obrazcka, & T. Turletti; “A survey of software-defined networking: Past, present, and future of programmable networks,” IEEE Communications Surveys & Tutorials Third quarter 2014, VOL. 16, no. 3, pp. 1617-1634; DOI: 10.1109/SURV.2014.012214.00180; Feb, 2014

[7] M. Raza, S. Sivakumar, A. Nafarieh, & B. Robertson; „ A comparison of Software Defined Network (SDN) Implementation Strategies,” Procedia Computer Science, VOL. 32, pp.1050-1055; June, 2014

[8] A. Wool, “A quantitative study of firewall configuration errors,” Computer, vol. 37, no. 6, pp. 62–67; 2004

[9] C. Vandana; “ Security Improvement in IoT based on Software Defined Neworking (SDN),” ISETR, VOL. 5, no. 1, pp.291-295; Jan, 2016

[10] Open Networking Foundation; “Software-Defined Networking: The New Norm for Networks,” www.opennetworking.org ; April, 2012

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson,J. Rexford, S. Shenker, and J. Turner; “OpenFlow: enabling innovation in campus networks”; ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, pp. 69–74; 2008

[12] B. A. Forouzan; "Data Communications and Networking," in Data Communication and Networking, Alan R. Apt, 2007, p. 90

[13] M. Rouse, "TechTarget," 2017. [Online]. Available: http://searchnetworking.techtarget.com/definition/round-trip-time. [Accessed 14 July 2017].

[14] T. Nadeau and P. Pan; “Software driven networks problem statement”; 2011

[15] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, “Towards a secure controller platform for OpenFlow applications”; in Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking. ACM, pp. 171–172; 2013

[16] M. Z. D. Hartman, S. Wasserman; “Security Requirements in the Software Defined Networking Model”; 2013

[17] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia; “Modeling and performance evaluation of an OpenFlow architecture,” in Proceedings of the 23rd International Teletraffic Congress. ITCP, pp. 1–7; 2011

[18] G. Yao, J. Bi, and L. Guo, “On the cascading failures of multicontrollers in software defined networks,” in Network Protocols (ICNP), 2013, 21st IEEE International Conference, pp.1-2; Oct 2013

[19] P. Fonseca, R. Bennesby, E. Mota, and A. Passito, “A replication component for resilient OpenFlow-based networking,” in Network Operations and Management Symposium (NOMS), 2012 IEEE, pp. 933–939; April 2012

[20] E. Al-Shaer and S. Al-Haj, “FlowChecker: Configuration Analysis and Verification of Federated OpenFlow Infrastructures,” in Proceedings of the 3rd ACM Workshop on Assurable and Usable Security Configuration,ser. SafeConfig’10. ACM, pp. 37–44; 2010

[21] T. Dierks, “The transport layer security (TLS) protocol version 1.2,” http://tools.ietf.org/html/rfc5246 ; 2008

[22] E. Rescorla and N. Modadugu, “Datagram transport layer security version 1.2,” http://tools.ietf.org/html/rfc6347 ; 2012

[23] K. Benton, L. J. Camp, and C. Small, “OpenFlow vulnerability assessment,” in Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN ’13. ACM, pp. 151–152; 2013

[24] R. Kandoi, and M. Antikainen; ”Denial-of-Service Attacks in OpenFlow SDN networks,” in 2015 IFIP IEEE international Symposium on

Integrated Network Management (IM), pp. 1322-1326. IEEE; 2015

[25] https://sourceforge.net/projects/sdn-toolkit/ ; July, 2016;

AUTHOR PROFILES:

Ayman Haggag was born in Cairo, Egypt in 1971. He received his B.Sc. degree from Ain Shams University, Egypt, in June 1994, M.Sc. degree from Eindhoven University of Technology, The Netherlands, in December 1997, and Ph.D. degree from Chiba University, Japan, in September 2008. Since March 1996, he has been with the Electronics Technology Department, Faculty of Industrial Education, Helwan University, Cairo, Egypt. His current research interests are in the fields of Networking, SDN, Wireless Sensor Networks, Network Security and Wireless Security.