A MODEL TO ENHANCE SECURITY IN SOFTWARE DEFINED NETWORKING (SDN)

SHAIK MOHAMMED SHAFIULLA

Assistant Professor,
Department of Computer Science & Engineering,
Scient Institute of Technology, Hyderabad,[India].

Abstract : Software Defined Networking, or SDN, is a new technology that has the potential to take the place of the usual vendor-based proprietary CLI networking devices.Applications-based network control has been introduced by SDN, which has presented numerous opportunities as well as challenges for research and innovation in these networks.Security is a concern for developers who want to invest in SDN, despite its numerous advantages and opportunities.I examine the SDN security issues and their solutions in this paper.I have developed a threat model for four different use cases that can be used to account for SDN's security needs.These are the use cases:I) safeguard controllers against applications; II) safeguard controllers between controllers; III) safeguard controllers against data plane or switches; and IV) safeguard controllers against malicious switches.If one of these SDN components is secure, another is already secure, as i discovered.In addition, i provided insights for protection mechanism and security enhancements by comparing SDN and traditional network security in relation to these four use cases.Based on the Ryu controller, a framework for creating an SDN security application has been presented.I believe that a ready reference for dealing with vulnerabilities and threats in this area will be provided by our threat model, which will assist numerous researchers and developers in comprehending the current security requirements.With our proposed security architecture, i conclude by identifying some unsolved research issues and potential future research directions.

Keywords : Software defined networking (SDN), openflow, control plane, data plane,controller; programmability

I INTRODUCTION

Traditional network (TN) devices, such as routers, switches, firewalls, load balancers, and so on, are extremely powerful and offer a variety of networking control functions.However, security is always a major concern because the network is distributed and contains a variety of devices that perform a variety of networking functions [1].Every year, a lot of new models are made with more processing power and new software versions from vendors, so customers have to buy new hardware to use the new software.These restrictive gadgets are expensive and have their own specific manner of arrangement through CLI, having a few explicit orders also, various sellers have various orders to

speak with these gadgets.There may be configuration errors and security breaches as a result [2].The results of these commands are as intended by the human operator, and further programmability cannot be achieved with this output.As a result, researchers and network engineers who want to scale and automate their network operations in response to demand cannot do so [3].Compared to system administration, where software is independent of the hardware, these hardware-dependent systems that are tightly coupled with software have failed to advance networking. An operating system is a piece of software that is independent of hardware in system administration.I am free to install any software and operating system on any hardware in accordance with the requirements.System administration is changing quickly as a result.Utilizing a hypervisor, which oversees multiple virtual machines running on distinct host operating systems, i am now able to install numerous servers on a single piece of hardware.Docker is another solution that provides high-level resource utilization [4], as shown in Fig.1 and 2, specifically.In the concept of virtual machines as depicted in Fig.1, a VM image that is used by a specific service receives dedicated processing resources and an operating system; however, Docker provides containers for hosting specific services or applications, which use very few resources compared to virtual machines, as depicted in Fig.2.On a single operating system, a single Docker engine can house thousands of containers running various applications on specific servers.In contrast, in network administration, I will continue to work with hardware-dependent networking devices that require a significant amount of processing power and time for manual configuration.The current networking architecture must be redesigned to meet the aforementioned requirements with automation, programmability, and flexibility.

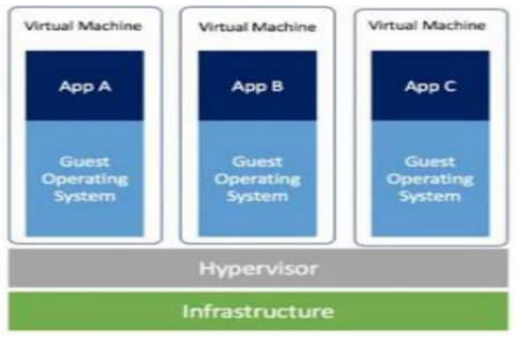


***Fig 1** : Virtual Machines hosted on Hypervisor*

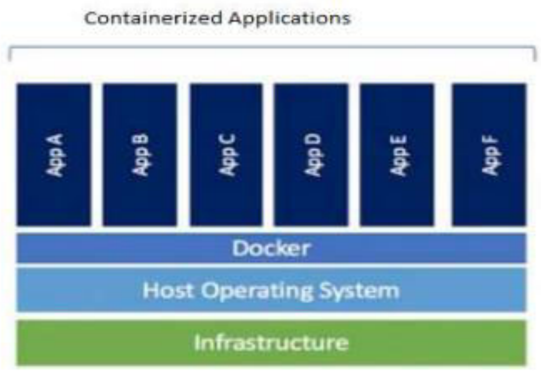


***Fig 2** : Containerized Applications on Single OS through Docker.*

Software Defined Networking [5] is a new concept which provides an API for configuration and decouples software logic from the devices. These devices work as simple data forwarding devices. The software or logical intelligence has been placed in a centralized controller. . The communication of forwarding devices and controller is established through a southbound API e.g. openflow [3]. All the networking functions like Routing, Security and Network monitoring etc. are done through the applications in application plane. The communication of application plane and controller is coordinated by northbound API e.g. RESTful API [6]. This provides the programmability approach and various applications can be designed a per the network demands. Network engineers can also use third party applications irrespective of hardware based solution for managing their network infrastructure. The idea of SDN is to use vendor specific hardware and I am free to choose software as per network demands irrespective of hardware. This arrangement of network functionality provides various opportunities for research and innovation in these networks. SDN is evolving and it has various advantages or traditional networks like dynamic control, programmability and a complete view of the network. As it is a new technology security solutions in SDN need to redefine and it provides various challenges and opportunities.

II THREAT MODEL

A threat model based on the SDN architecture depicts the various ways in which SDN components can be attacked.If one component is compromised, SDN components are interconnected.It poses a threat not only to one component but also to the entire network.Identifying the various attacks that an attacker could use against a specific SDN component is our objective here.SDN applications are in the Application Plane, controllers are in the Control Plane, and networking devices like switches are in the Data Plane.In Fig.3 The SDN block diagram with its components has been shown.I have derived four threat analysis use cases from this architecture.

SDN security issues and solutions can be displayed in a variety of ways [10][11].The majority of authors talk about the same thing with a layer-based approach, but i think that SDN architecture is different from a traditional network in some ways, so i created a new taxonomy to cover all SDN security issues. A network scenario with n no.of regulators

C = {c1,c2, … … .cn}.From the set of applications Aci = a1.a2,......an, each controller ci C can run at least one application.Due to their limited resources, each controller is susceptible to denial-of-service attacks.Four use cases have been derived from SDN architecture.The security objectives and importance of each use case vary.Fig.4 depicts the Threat model for the SDN's security requirements.Figure depicts the SDN architecture and associated use cases.3 and 4 in particular. A passive attack known as a "semi-beneficial" attack may collect information about the network or processes, but it will not alter the protocol's execution. An active threat that deviates from protocol rules in order to disrupt the system and attack other components of the system is referred to as malevolent behavior [12][13].The four use cases are outlined below.

2.1 Use Case 1: Securing Controller from Applications in Application Plane

In this use case, every application in the application plane can be good, half-good, or bad.These applications might come from third-party apps, for example [14].The controller provides an abstraction to the application plane so that the application can generally read and edit network state, which is a form of network control.An attacker can impede network operations by impersonating an application and gaining access to controller, or network control [15].Spoofing attacks may result from an absence of trust and inadequate authentication between controllers and applications [16][17].The reduction of applications' attacks on controllers is our objective here.

2.2 Use Case 2: Inter Controller Security

In SDN, control is logically centralized. It provides more than one controller for providing scalability and avoiding single point of failure [18]. As a result these controllers share the resources and communicate with each other. It is necessary to review the security of inter controller communication [19]. In this use case i assumed one or more controller is semi benign or malevolent. A semi benign controller could be able to access the control data of other controllers, learn resource utilization information and target the integrity of the network. Moreover a malevolent controller can attack to semi benign controller and perform a DoS attack on another controller. Our goal is to protect controller from each other [20].

2.3 Use Case 3: Securing Switches from Controller

In this use case it is assumed at least one controller is semi benign or malevolent. I assumed that applications which are used through this controller can be semi benign or malevolent.A semi benign controller can target switches in the data plane. It can attack switch flow table with buffer overflow by sending bogus entry [21]. Our goal here is to eliminate the possibility of controller's ability to target the switch with bogus entry [22].

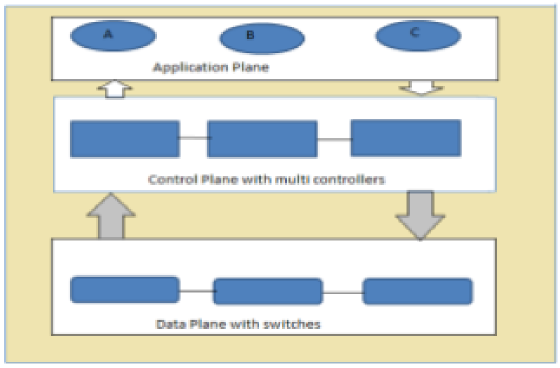
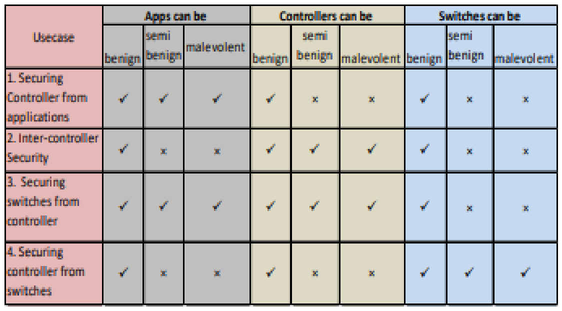


Fig 3 : SDN Architecture



Usecase	Apps can be			Controllers can be			Switches can be		
	benign	semi benign	malevolent	benign	semi benign	malevolent	benign	semi benign	malevolent
1. Securing Controller from applications	✓	✓	✓	✓	×	×	✓	×	×
2. Inter-controller Security	✓	×	×	✓	✓	✓	✓	×	×
3. Securing switches from controller	✓	✓	✓	✓	✓	✓	✓	×	×
4. Securing controller from switches	✓	×	×	✓	×	×	✓	✓	✓

Fig. 4. *Threat Model*

III SECURITY ENHANCEMENTS IN SDN

Four of the most significant attack use cases in SDN and traditional networks have been compared.In contrast to controller attacks in SDN, I have seen how control functions in TNs can be attacked in a variety of use cases [40]. The lessons learned from comparing these use cases in terms of threats and defenses will now be discussed.Based on our threat model, i will investigate how SDN security can be improved [41].Before developing a security application based on attacks from the aforementioned use cases, i will first elaborate on each use case.

3.1 Usecase 1 :

Securing Controller from Applications Since network control functions are part of network devices in traditional networks, this situation does not apply to TNs, as i discussed in the previous section.Decoupled from network devices, network control functions in SDN take the form of applications.These applications work with the controller and data plane devices [42].In point of fact, these applications communicate with the controller in order to fulfill the requirement of the network. An unauthorized application, on the other hand, has the potential to significantly harm the controller and even reconfigure the network [43].Before exchanging control messages, the access controller and the application need to maintain a trusted connection in order to defend against an unauthorized application.Before establishing a connection, applications must be validated for both authentication and authorization.[44] discusses this concern regarding the controller's authentication and security by untrusted applications.A controller hierarchy was introduced by the authors.Because the application's code runs in the middle of the hierarchy, where there is a lot of protection, this hierarchical system can reduce the impact of harmful applications.FortNox [45] is another piece of work in this direction.The open-source controller NOX is the foundation for FortNox [29].It is a security enforcement kernel that monitors the flow rules in real time for security policy violations.A role-based authentication approach grants authorization to each openflow application.The roles of flow rule producers are as follows:OF Security, OF Application, and OF Operator.A higher priority rule is accepted if FortNox finds a flow rule conflict.Application identification and priority enforcement are FortNox's limitations.

The improvement to the controller's resistance to malicious applications is ROSEMARY [46].It is a robust and secure network operating system with high performance.Each application instance that is running is sandboxed to protect the control layer from any vulnerabilities.Additionally, it regulates and monitors the resources utilized by each application.The authors of LegoSDN [47] investigate the impact of application failure on controller reliability.The authors suggested putting an isolation layer between the controller and the applications to prevent the controller from failing because of an application failure.

3.2 Usecase2 :

Multiple controllers have been suggested for inter-controller protection in SDN in order to prevent a single point failure.There are two types of placement schemes for controllers:Both flat controller deployment and hierarchical controller deployment are options.In the flat controller idea, each controller gets its own subnet.Different operations may not be able to communicate equally with various domains in this solution.However, in hierarchical mode, the global controller is in charge of the local controller and the local controller is in charge of the respective network.The global controller is the medium through which the various controllers communicate.The controller placement issue has been addressed through a number of different efforts.An algorithm for determining the optimal controller load and minimum number of controllers was proposed in [48].However, for the request with variable time, this arrangement was ineffective.The author of [49] proposed an algorithm that divides the network into multiple subnets.Each little organization contains a regulator in view of the size of relegated network.It divides the network according to switch density using a clustering algorithm.It may use a backup link in the event that the primary link fails.However, it might cause an unnecessary delay.A multi-controller solution with a Byzantine fault-tolerant mechanism is presented by the authors in [50].When one controller fails, the network is managed by the other controller, which also removes the previous controller's idle link.However, due to performance issues in larger networks, this solution works well for smaller networks.

3.3 Usecase 3 :

Protecting Switches from Controllers In SDN, the controlling element controller has more functionality, making it possible for a malicious controller to cause significant damage to data plane switches.By generating broadcast that isn't needed, a compromised controller can attack the switch flow table and overflow the table.Thus, the primary defense for data plane switches is to prevent malicious activity from occurring on the controller.A method for spotting a malicious SDN device in the network was suggested by the authors in [26].They set up a backup controller and gathered state updates and information from the primary controller and switches.By recognizing the primary controller, backup controller, and SDN switches' unexpected and inconsistent behavior, they identify malicious devices.

By the comparisons and discussion in the last two sections it can be stated that there is a need to develop a security mechanism to counter the security issues of SDN. As discussed that the controlling functions in the SDN are performed by the applications in application plane. For implementing the security functions there is a need to design the security application in SDN. But this is advancement in SDN that network controlling functions like security, routing, and monitoring etc., are in the form of applications. For Design and implementation, I used mininet as network emulator and Ryu as a controller. First i will focus basic steps and algorithm for designing an application as per controller and data plane communication.Python language is used to develop the network applications based on Ryu controller. Ryu is a components based controller which has various modules for application design and control. In ryu controller setup at home/ubuntu/ryu it has various folders; app, base and ofproto. App folder can contain various applications like firewall, router and load balancer. Base folder contains

App_manager which helps to run the different applications and prepares framework and datapath for running the application. Ofproto deals with openflow version related queries and matching capabilities. For designing a SDN application need to collect and understand the initial requirements and booting process of SDN network framework.

a)In first step switch boots up and contact the controller for openflow version related queries and check its capabilities.

b)The controller installs Packet In function and table miss function and prepares itself for queries from switch.

c)When receiving Packet In, Controller learns the source MAC and mention the MAC and port information in flow table. It checks for destination MAC address if it is available in flow tables, it uses Packet Out function on the port and installs the flow and stores the same for future uses.

d)If destination MAC address is not available in flow table i.e. a table miss then controller uses packet out function to broadcast the packet to all ports.

By using the ryu controller framework i can design and deploy customized security applications. With programmability approach in SDN , i can have our own security application in ryu app folder and program it as per network demands and configure it through standard API. Traditional security solutions, the vendor specific e.g. fortigate and Cisco, they have their own proprietary code and configuration methods which are fixed and cannot be customized as per demands. When Host A wants to communicate to Host B it sends a packet to switch. Switch check for a matching entry in its flow table but when a matching entry is not found in flow table then packet is forwarded to controller. Controller sends the packet to security application for policy check. First it parses the packet and check if it matches to policy specified in firewall. As firewall has a policy to block traffic from A to B (A-->B: Block). The application enforces a rule through controller to drop the packet and controller install a flow rule in switch flow table to drop all the incoming traffic from Host A to Host B. This is how i can block and allow flow in openflow through a security application. It means through this app a switch can work like a firewall i.e. technology allows us to decide the functions of a switch. As a result additional security devices are not required in SDN as security services can be enabled within the devices. In traditional network another problem is placement of firewall for optimized coverage of security services. But it has been nullified as any device in the network can be turned into a security device.

IV CONCLUSION

i created four use cases and presented a tabular discussion of several attack parameters and their countermeasures to identify SDN security issues.I used the same use cases from the traditional network after identifying the security issues for a comparison of risk and security technology in both networks.Comparative research has led researchers to the conclusion that SDN has provided traditional networks with new attack surfaces.SDN, on the other hand, gives you more control over the network, automation, and flexibility than traditional networks do.However, security solutions have been presented to address SDN security issues, including protection from malicious applications, protection of the data plane, protection from DoS attacks by data plane switches, and protection of the controller.Ryu controller and mininet network emulator are included in the framework that has been presented for the development of an SDN security application based on analysis.A proposed security model that is based on recent research and threat model analysis has been presented to provide insights for improving security.Furthermore, research into SDN security is still in its infancy, and there is still much to be done.I can find superior SDN networks that will be significantly more secure than traditional networks by developing novel security techniques and expanding on previous research to address known issues.

REFERENCES

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

[2] M. Casado et al., “Ethane: Taking control of the enterprise,” in ACM SIGCOMM Comput. Commun. Rev., vol. 37, no. 4, pp. 1–12, Oct. 2007.

[3] N. McKeown et al., “OpenFlow: Enabling innovation in campus networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Apr. 2008.

[4] R. R. Yadav, E. T. G. Sousa and G. R. A. Callou, “Performance Comparison between Virtual Machines and Docker Containers”, IEEE Latin America Transactions, VOL. 16, NO. 8, AUG. 2018, pp. 2282-2288.

[5] S. Jain et al., “B4: Experience with a globally-deployed software defined

WAN,” in Proc. ACM SIGCOMM Conf., 2013, pp. 3–14.

[6] Li Li, Wu Chou, Wei Zhou and Min Luo, “ Design Patterns and Extensibility of REST API for Networking Applications”, IEEE, TNSM, 2015, 00814.

[7] S. Taha Ali et. al “A Survey of Securing Networks using SDN”, IEEE transactions on reliability, Vol 64, No. 3, 2015.

[8] Marc C. Dacier et al, “Security Challenges and Opportunities of Software Defined Networking”, in IEEE Computer and Reliabilities Societies, 2017, pp.96-100.

[9] B. Ahmad et al. “Fingerprinting SDN policy parameters : An Empirical Study”, IEEE Access, Volume 8, 2020.

[10] D. Li, X. Hong, and J. Bowman, “Evaluation of security vulnerabilities by using ProtoGENI as a launchpad,” in Proc. IEEEGLOBECOM, 2011, pp. 1–6.

[11] S. Shin and G. Gu, “Attacking software-defined networks: The first feasibility study,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 165–166.

[12] L. Schehlmann, S. Abt, and H. Baier, “Blessing or curse? Revisiting security aspects of software-defined networking,” in Proc. 10th Int. CNSM, 2014, pp. 382–387.

[13] S. Sezer et al., “Are we ready for SDN? Implementation challenges for software-defined networks,” IEEE Commun. Mag., vol. 51, no. 7, pp. 36–43, Jul. 2013.

[14] W. Han, H. Hu, and G.-J. Ahn, “LPM: Layered policy management for software-defined networks,” Data and Applications Security and Privacy XXVIII. Berlin, Germany: Springer-Verlag, 2014, pp. 356–363.

[15] X. Wen, Y. Chen, C. Hu, C. Shi, and Y. Wang, “Towards a secure controller platform for OpenFlow applications,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 171–172.

[16] S. Scott-Hayward, C. Kane, and S. Sezer, “OperationCheckpoint: SDN application control,” in Proc. 22nd IEEE ICNP, 2014, pp. 618–623.

[17] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, Securing the software-defined network control layer,” in Proc. NDSS, San Diego, CA, USA, Feb. 2015, pp. 1–15.

[18] P. Berde et al., “ONOS: Towards an open, distributed SDN OS,” in Proc.3rd Workshop Hot Topics Softw. Defined Netw., 2014, pp. 1–6.

[19] M. M. O. Othman and K. Okamura, “Securing distributed control of software defined networks,” Int. J. Comput. Sci. Netw. Security, vol. 13, no. 9, pp. 5–14, Sep. 2013.

[20] F. Botelho, A. Bessani, F. M. Ramos, and P. Ferreira, “On the design of practical fault-tolerant SDN controllers,” in Proc. 3rd EWSDN, 2014, pp. 73–78.

[21] H. Mai et al., “Debugging the data plane with anteater,” ACM SIGCOMM Comput. Commun. Rev., vol. 41, no. 4, pp. 290–301, Aug. 2011.

[22] Ahmad, B. et al., “Fingerprinting SDN policy parameters : An Empirical Study”, IEEE Access, Volume 8, 2020.

[23] C. Jeong, T. Ha, J. Narantuya, H. Lim, and J. Kim, “Scalable network intrusion detection on virtual SDN environment,” in Proc. IEEE 3rd Int. Conf. CloudNet, 2014, pp. 264–265.

[24] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting traffic anomaly detection using software defined networking,” in Recent Advances in Intrusion Detection. Berlin, Germany: Springer-Verlag, 2011, pp. 161–180.

[25] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” in Proc. IEEE 35th Conf. LCN, 2010, pp. 408–415.

[26] J. Suh et al., “Implementation of content-oriented networking architecture (CONA): A focus on DDoS countermeasure,” in Proc. European NetFPGA Developers Workshop, Cambridge, U.K., 2010, pp. 1–6.

[27] Purnima Murali Mohan et. al., “Towards resilient in-band control path routing with malicious switch detection in SDN”, IEEE COMSNETS, 2018, PP.9-16.

[28] Haifeng Zhou et. al., ”SDN-RDCD: A Real-Time and Reliable Method for Detecting Compromised SDN Devices”, IEEE/ACM transactions on networking, vol. 26, no. 5, october 2018 pp. 2048-2061

[29] D. Kreutz, F. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 55–60.

[30] S. Shin et al., “FRESCO: Modular composable security services for software-defined networks,” in Proc. Netw. Distrib. Security Symp., San Diego, CA, USA, 2013, pp. 1–16.

[31] N. Gude et al., “NOX: Towards an operating system for networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 3, pp. 105–110, Jul. 2008.

[32] D. Erickson, “The beacon OpenFlow controller,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 13–18.

[33] A. Guha, M. Reitblatt, and N. Foster, “Machine-verified network controllers,” ACM SIGPLAN Notices, vol. 48, no. 6, pp. 483–494, Jun. 2013.

[34] S. H. Yeganeh and Y. Ganjali, “Kandoo: A framework for efficient and scalable offloading of control applications,” in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 19–24.

[35] N. Foster et al., “Frenetic: A network programming language,” ACM SIGPLAN Notices, vol. 46, no. 9, pp. 279–291, Sep. 2011.

[36] T. Koponen et al., “Onix: A distributed control platform for large-scale production networks,” in Proc. OSDI, 2010, vol. 10, pp. 1–6

[37] Seung Yeob Nam, Dongwon Kim and Jeongeun Kim. “Enhanced ARP: Preventing ARP Poisoning-Based Man-in-the-Middle Attacks” IEEE Communications Letters, Vol. 14, No. 2, February 2010, pp. 187-189.

[38] Songyi Liu, “MAC Spoofing Attack Detection Based on Physical Layer Characteristics in Wireless Networks” IEEE, ICCEM, 2015.

[39] Timo Kiravuo, Mikko S¨arel¨a, and Jukka Manner, “A Survey of Ethernet LAN Security” IEEE Communications Surveys & Tutorials, Vol. 15, No. 3, 2013,pp. 1477-1491.

[40] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, “OrchSec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions,” in Proc. IEEE NOMS, 2014, pp. 1–9.

[41] Pradeep Kumar Sharma and S.S Tyagi “Improving Security through Software Defined Networking (SDN): An SDN based Model”, IJRTE, vol. 8, issue 4, 2019, pp. 295-300.

[42] Marcelo Ruaro , Luciano Lores Caimi , and Fernando Gehm Moraes, “SDN-Based Secure Application Admission and Execution for ManyCores”, IEEE Access, volume 8, 2020, pp. 177296- 177306.

[43] D. Kreutz et al., “Software-defined networking: A comprehensive survey,” arXiv preprint arXiv:1406.0440, 2014.

[44] D. Yu, A. W. Moore, C. Hall, and R. Anderson, “Authentication for resilience: The case of SDN,” in ser. Security Protocols XXI. Berlin, Germany: Springer-Verlag, 2013, pp. 39–44.

[45] P. Porras et al., “A security enforcement kernel for OpenFlow networks,” in Proc. 1st Workshop Hot Topics Softw. Defined Netw., 2012, pp. 121–126.

[46] S. Shin et al., “Rosemary: A robust, secure, and high-performance network operating system,” in Proc. ACM SIGSAC Conf. Comput. Commun. Security, 2014, pp. 78–89.

[47] B. Chandrasekaran and T. Benson, “Tolerating SDN application failures with LegoSDN,” in Proc. 13th ACM Workshop Hot Topics Netw., 2014, p. 22.

[48] G. Yao, J. Bi, Y. Li, et al., “On the Capacitated Controller Placement Problem in Software Defined Networks”, IEEE

Communications Letters,vol.18, no.8, 2014, pp. 1339-1342.

[49] J. Liao, H. Sun, J. Wang, et al., “Density cluster based approach for controller placement problem in large-scale software defined

networkings”, Computer Networks, vol.112, 2017, pp. 24-35.

[50] H. Li, P. Li, S. Guo, et al., “Byzantine-resilient secure software-defined networks with multiple controllers”, Proc. IEEE International Conference on Communications, 2014, pp. 695-700.

[51] OpenFlow Switch Specification Version 1.4, Open Network.

[52] K. Benton, L. J. Camp, and C. Small, “OpenFlow vulnerability assessment,” in Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw., 2013, pp. 151–152.

[53] Josy Elsa Varghese and Balachandra uniyal, “An efficient IDS framework for DDOS attacks in SDN environment”, IEEE Access, 2021, pp. 69680-69699.

[54] Tooska Dargahi et. al., “A Survey on the Security of Stateful SDN Data Planes” IEEE Communications Surveys & Tutorials, Vol. 19, No. 3, 2017, PP. 1701-1724.

[55] A. A. Z. SOARES et. al., “3AS: Authentication, authorization, and accountability for sdn-based smart grids ”, IEEE Access, volume 9, 2021, pp. 88621-88640

[56] Kevin Barros Costa et al., “Enhancing Orchestration and Infrastructure Programmability in SDN with NOTORIETY”, IEEE Access, Volume 8, 2020, pp. 195487-195502.

[57] Basem Almadani , Abdurrahman Beg and Ashraf Mahmoud, “DSF: A Distributed SDN Control Plane Framework for the East/West Interface” IEEE Access, Volume 9, 2021, pp. 26735-26754.

[58] Ahmed Sallam , Ahmed Refaey,and Abdallah Shami, “On the Security

of SDN: A Completed Secure and Scalable Framework using the Software-Defined Perimeter”, IEEE Access, volume 7, 2019. pp. 146577-146587