JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Security Overview of Software Defined Networks: Threats and Countermeasures

[1]Ayman Haggag, [1]Hoda Youssef, [2]Ihab Ali, [2]Fatty M. Salem

[1]Electronics Technology Department, Faculty of Technology and Education, Helwan University, Cairo, Egypt
[2]Electronics and Communications Engineering, Department Faculty of Engineering, Helwan University, Cairo, Egypt

Abstract : Software-Defined Networking (SDN) is a computer network that promises to enhance the infrastructure and architecture of networks. SDN depends mainly on decoupling the control plane from the data plane. In SDN, the control plane is implemented outside the forwarding element (OpenFlow switch). The SDN controller is considered the basic component in the SDN architecture, which can represent the control plane of a network. Hence, SDN has many advantages over traditional networks due to the centralization of a network's control plane in addition to supporting network programmability. SDN is capable of enhancing network manageability, configuration, and troubleshooting. However, SDN can suffer from many network security attacks and threats. Therefore, protecting the SDN architecture from attacks is a demanding and urgent issue. Hence, in this research, we review security requirements for in SDN and countermeasures that defend the SDN controller.

Index Terms **– Network Security, Software Defined Networking, SDN Structure.**

I. INTRODUCTION

Recently, there was a need to address the difficulties of traditional networks faced by many of its users, and there was a call to adopt new approaches to communication in order to address these difficulties. The solution to these problems is represented in a new structure in the world of networks, which is SDN Networks, which is a real development in the world of networks, where SDN simplifies network management and its ability to be programmed, improve networks, and the possibility of innovation and flexibility. It is also characterized by the central control feature, by separating the control plane from the data plane [1]. As a result, SDN designs have two major differentiating characteristics [2, 3].

1.1 Control Plane and Data Plane Separation

Separating both the control plane and the data plane, where the control plane is representing all the basic devices of the network, called the controller, leads to reducing cost and reducing network complexities. The controller is responsible for managing the network and controlling it more easily and quickly. The network administrators do not need to configure or manage the network devices separately. SDN networks also enjoy a supposedly vision and global knowledge [3].

1.2 Programming Interfaces

SDN Networks add powerful and convenient programming interfaces that facilitate the design and configuration of various network applications and programs e.g., traffic engineering and quality of service applications, and using basic programming commands and lines of code, so it is easy to deploy. In SDN networks, any new configurations can also be added and applied within the network, and errors can be discovered and repaired, due to the network's flexibility and programmability [3, 4]. Therefore, the features of SDN, in addition to its architecture, benefits, threats, and countermeasures will be described in this paper [5]. The SDN weakness point is the SDN controller as it may be a single point of attack that compromises the whole network.

1.3 Article Organization

This paper is organized as follows, Section I contains the introduction of this research, Section II contain the SDN architecture, Section III contains the benefits of SDN, Section IV contain SDN threats and countermeasures, section V concludes research work with future directions.

II. SDN ARCHITECTURE

SDN design has three layers, as indicated in Figure 1: infrastructure layer, control layer, and application layer.

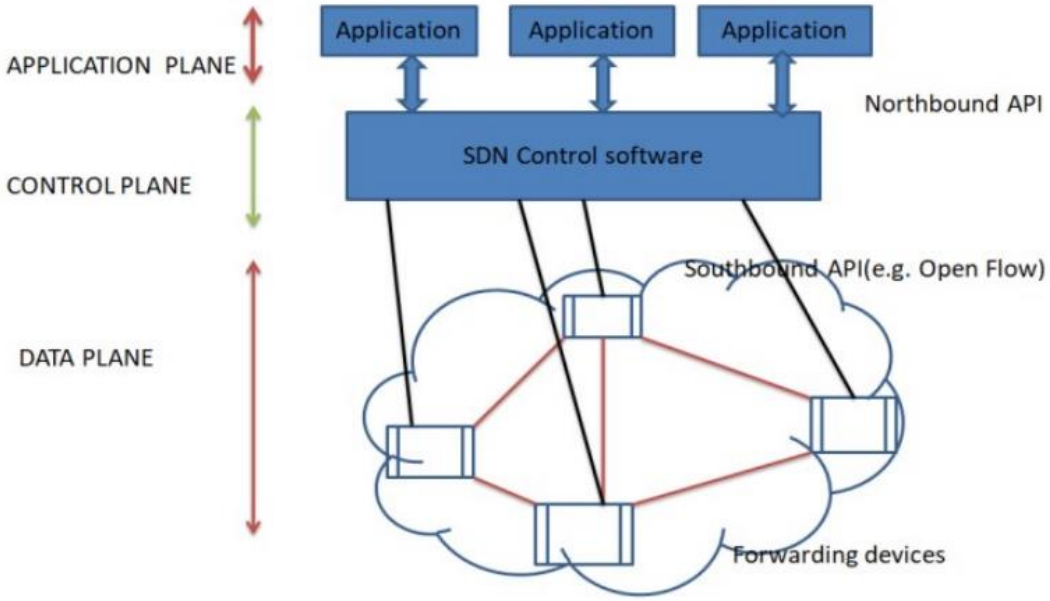

Figure 1. Software-defined networking architecture

The following are the six key components of SDN architecture:

2.1 Management Plane

The first is the management plane, for starters, which is a collection of SDN network applications that govern and control the logic. SDN enables network programmability, allowing for faster and more flexible deployment of new services and applications. such as load balancing and routing. It also ensures network automation and orchestration with existing APIs [6].

2.2 Control Plane

Second, the control plane is a basic layer in control plane elements and programmable forwarding devices are controlled by the SDN architecture through well-defined SI embodiments which are responsible for forwarding devices in SDN networks where the control plane contains the controller and applications where all the control logic resides [6].

2.3 Data Plane

The third element is the Data plane including wireless radio channels interconnect forwarding device. The data plane is the network infrastructure which is a representation of interlinked forwarding devices.

2.4 Southbound Interface

Fourth, the interface between the control plane component and forwarding devices is known as the southbound interface. The data plane's connection with the control element is formalized by the communication protocol.

2.5 Northbound Interface

The fifth API is the northbound API which allows communication between the management and control layers via northbound interfaces. The network topology, provisioning tasks, and configuration retrieval are all available through this northbound interface. Through the provisioning northbound API, it also provides network, loop avoidance, path calculation, security, and routing, as well as fault management.

2.6 Forwarding Devices

The sixth component is the forwarding devices which are either hardware or software-based, conduct a series of actions through data plane devices. The forwarding devices enable the use of a well-defined instruction set to operate on incoming packets. The southbound interface defines these instructions. The protocol instructions are also defined in these instructions, which are installed by the SDN controllers in forwarding devices to implement the southbound protocol.

The Open flow was normalized by the leading IT industry like Facebook, Cisco, Google, HP, etc. Thus, Understanding OpenFlow architecture contained in the following subdivision needs knowing the SDN notation. Opflex [7] extends the complexity of managing networks to advance the scalability of the infrastructure layer [7]. Management of conventional networks is enhanced by presenting a flexible technique without using a logically centralized controller. Whereas ROFL [8] gives an API relying on the open flow that allows full development for software developers for the new application.

III. SDN Benefits

With intrinsic decoupling of the control plane from data plane, SDN enables more network control through programming. Table 1 summarizes the possible benefits of this combined feature, which include simplified configuration, higher performance, and promoting creativity in network architecture and operations. SDN, for example, may integrate not just packet forwarding at the switching level but also link tuning at the data link level, breaking the layering barrier. SDN also enables real-time centralized network control on the basis of both instantaneous network state and user-defined policies, thanks to its ability to collect immediate network status. This has the added advantage of permitting network settings to be optimized and network performance to be improved. SDN's potential benefit is further shown by the fact that its network programmability and ability to define isolated virtual

networks via the control plane made of it a suitable platform for experimenting with new approaches and encouraging novel network designs, as detailed in table 3.1.

Table 3.1: Comparisons between SDN and Conventional Networking [9]

	SDN	Conventional Networks
Features	Decoupled data and control plane, and programmability.	A new protocol per problem, complex network control.
Configuration	Automated configuration with centralized validation.	Error prone manual configuration.
Performance	Dynamic global control with cross-layer information.	Limited information, and relatively static configuration
Features	Decoupled data and control plane, and programmability.	A new protocol per problem, complex network control.
Innovation	Easy software implementation for new ideas, sufficient test environment with isolation, and quick deployment using soft-ware upgrade.	Difficult hardware implementation for new ideas, limited testing environment, long standardization process.

In this section, we dwell on these benefits of SDN [9].

3.1 Improving Configuration

One of the most crucial responsibilities in network administration is configuration. To achieve consistent network operation, appropriate configurations are wanted when new equipment is introduced to an existing network. However, because of the disparity in network device manufacturers and configuration interfaces, most modern network configurations require some user intervention. This manual configuration process is time-consuming and prone to errors. At the same time, troubleshooting a network with configuration issues takes a lot of time and effort. It is widely acknowledged that, with the current network design, automatic and dynamic network reconfiguration remains a significant difficulty. SDN will support remedying such a situation in network management [9].

3.2 Performance Improvement

Current approaches frequently concentrate on development, improving the performance of a subset of networks or the user experience for specific network services. Obviously, these approaches, which are focused on local data and do not take cross-layer considerations into account, could result in weak inferior performance, if not conflicting network processes. SDN's arrival provides a global chance to increase network performance. SDN, in particular, allows centralized control with a global network perspective as well as feedback control with data transferred across levels in the network design. As a result, with appropriately designed centralized algorithms, many difficult performance optimization challenges would become doable. As a result, innovative solutions for classic problems like data traffic scheduling, end-to-end congestion control, load-balanced packet routing, and Quality of Service may be developed and implemented quickly to test their efficacy in enhancing network performance. [9].

3.3 Encouraging Innovation

Encouraging Innovation: due to the continuous change, future networks should support innovation rather than attempting to completely foresee and perfectly match future application requirements. Unfortunately, any new idea or design meets immediate implementation, experimentation, and deployment hurdles in existing networks. The biggest stumbling block is the widespread use of proprietary hardware in traditional network components, which prevents experimentation. Although experiments are possible, they are frequently carried out in a separate, simplified testbed. These tests do not provide enough assurance that these novel concepts or network designs will be adopted in the industrial sector. The concept underpinning community attempts to enable large-scale experiments cannot fix the problem [9].

On the other hand, SDN encourages innovation by supplying a programmable network platform on which new ideas, new applications, and new revenue-generating services may be easily and flexibly implemented, experimented with, and deployed. SDN's high configure allows for apparent separation of virtual networks, allowing for real-world testing. A seamless transition from an experimental phase to an operational phase can be used to launch innovative concepts in stages [9].

IV. SDN Security Threats and Countermeasures

The Software-Defined Networking architecture is now being employed in the provision of speedier services due to its advantages. SDN, like many other technical paradigms, does, however, pose security risks, some of which are exacerbated by its layered design [10].

4.1 Security of Northbound Interface and Application Plane

There is a collection of programs requirements and abstracts provided by the application plane which is essential to meet the specific system requirements allowing response to the SDN environment's criteria through the controller. The set of existing applications such as firewalls, routing policies, protocols, and many more, which the controller's developer can offer or third parties.

Communication occurs between the application plane and the controller through the northbound interface and is not standardized so each controller defines its own raw APIs and the type to be used e.g., such as RESTful APIs, programming languages, and customized APIs (ad hoc) [10][11].

REST is currently the industry's most of the applications extensively utilized API, but ONF is working on an open-source API as it does not rule out the possibility of standardization and tasks into account all stakeholders' participation (e.g. researchers, controller providers, application developers) and is involved in a more Efficiency in creating applications [10].

SDN's various applications are frequently implemented by third parties, which exposes them to security threats as shown in Figure 2, where a malicious application can masquerade as a real application and enter the controller and introduce wrong rules that modify network performance [12]. Specifically, Various current applications have conducted their security analyses to identify the weaknesses that cause attacks [13]–[17], which in turn reduce the performance efficiency in SDN networks, and from the research that helped reduce these attacks, for example, Indago [18] where malicious applications are detected proactively utilizing security-sensitive behavior graphs (SSBGs) graphs and machine learning. the authors focus on attacks that include tampering with information, masquerading as real users, and appropriating permissions, all of which disrupt network services. As for the proposal, it is capable of detecting interference through algorithms used in the same code of SDN's application MSAID [19], SAIDE [20] another proposal also that someone detects the interference using mathematical models and gets rid of it [10].

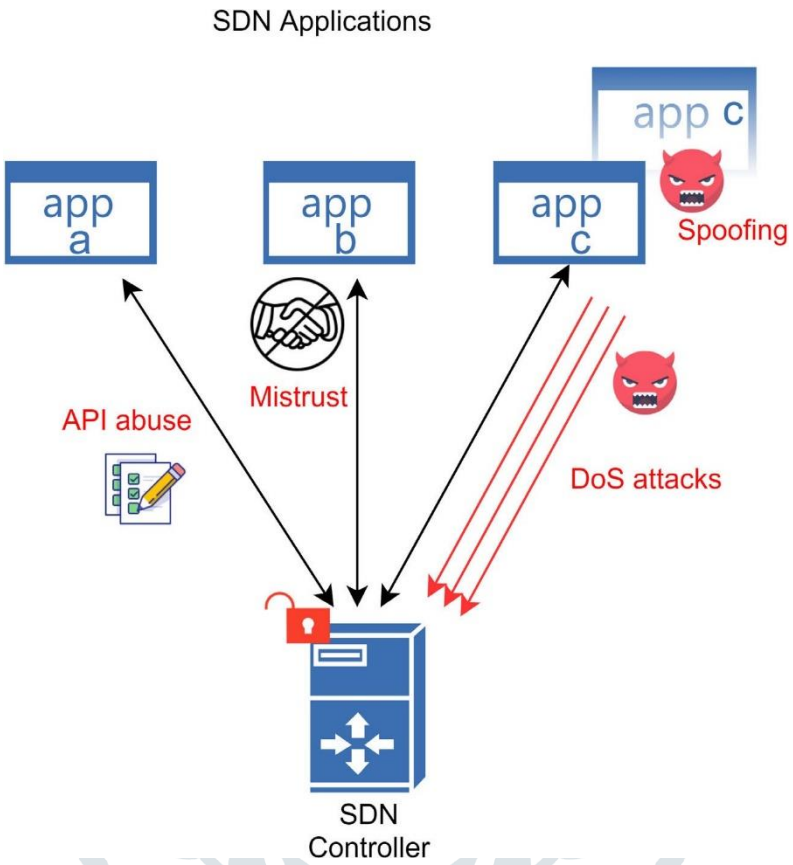

Figure 2. Associated Attacks emerging from applications [10]

Through these introductions, we find that the absence of effective and real previous authentication mechanisms, access control, and validation of SDN applications allows the northbound interface to implement trust relationship attacks against the controller and impersonation. In the absence of prior effective mechanisms that deal with the attacks that SDN networks are exposed to SDN applications are exposed to dealing with illegal requests that lead to resource exhaustion and poor network performance. In the proposal [21], after performing the performance tests, stress tests, and DDoS/DoS tests from the northbound facade, it was found that there is a weakness in the performance of the analyzed controllers. fine-grained and coarse-grained access controls are used to contribute to solving application authorization problems, where they are used to cover vulnerabilities of a standalone application or system, where it is a persistent environment where normal behavior is highly predictive. As for dynamic environments, there are fine-grained access controls with high accuracy and more detail in the application permissions [10].

Generally, In SDN networks, many services, providers, and multi-domain environments are used. The SDN architecture is also used in multi-tenant environments. Therefore, incorporating very strict coarse-grained access control may lead to the abuse of permissions or disruption of some network tasks. In the proposal [22] the authors explain that there is a loose coarse-grained access control mechanism to maintain network security from integrity attacks on the information flow as it allows the so-called application poisoning attack which one application can control to do tricks on other applications and perform actions. From the previous argument, we find that SDN applications authorization issues are resolved through access control. One solution is to use standalone mechanisms that rely on existing controllers to solve authentication problems. With this background, the authors have made the following contributions. In [23], the authors propose MD-UCON, which is an access control mechanism for the northbound interface, focusing on multidomain scenarios to facilitate cross-domain access control, this proposal leverages RBAC techniques that have been developed for dynamic situations, as well as a cross-domain role mapping approach. This solution is based on a model named UCON [10].

The authors in [24] proposes leveraging identifiers in the authentication and privilege assignment process to link controllers with OpenFlow apps via blockchain. This approach, in addition to providing Authentication, authorization, and accounting (AAA), provides decentralization and immutability of the database in which application information, permissions, and tokens are kept; it

also permits control of logs, which record all participants' behavior and allow monitoring. To avoid the exploitation of static permissions provided to an API, the authors in [25] proposes a controller-independent approach. It consists of a northbound security extension that serves as a "mediator" between the OpenFlow applications and the controller by repackaging the built-in services of the northbound APIs. This approach also includes controller-specific IDS, which stores information about the permissions and accounting records of the OpenFlow application, and a high-level policy engine predefines the policies for each OpenFlow application. In this way, OpenFlow applications can be authenticated and authorized, and the legitimacy of accounting requests can be verified using password-based authentication and token-based authentication. There is a security rule generator, which defines the access rules between applications, APIs, and their inputs or outputs. Finally, this frame features a decision engine that employs API hooking to intercept application activity. Here, the input/output of each API is reviewed and contrasted with the rules.

Also, some proposals and solutions grant permissions to third-party applications such as BEAM [26], which depends on the network behavior taken from metrics such as flow_injection_rate or packet_in_rate, which is analyzed by an IDS, and then the permissions of the applications are upgraded or reduced at runtime. BEAM has units it works with the registration wizard, which is responsible for new applications, as it registers and allocates initial permissions to them; Policy engine: It consists of two bases, namely, the policy store and the map table, the activity discovery unit and the activity engine, which are units responsible for reviewing the application activity that is benefited from it on IDS and the records, respectively [10].

Because of the presence and injection of malicious applications, misuse of the API and DoS, a structure called SENAD has been proposed that consists of four parts. Where the interaction between the application-plane controller and the data-plane controller is done by controlling factors based on the publication/subscription model. The resources of each application are controlled by the policy engine and also access control for them. Applications are isolated by the controller, application sandboxes, and resources are served as required by the policy engine. Finally, both the authentication and authorization work with the information exchanged between the APC and the DPC. The authentication is based on the password and rules that are consulted with the policy engine for authentication. [28], an approach called Application Authentication System deployed outside the console enabling authentication, log unauthorized operations, access rights, resources, application certificates, authorization components, encryption, and authentication are just a few of the things that can be managed.

Also, in [29], a security-as-a-service (SEaaS) solution is defined. Using a floodlight controller as a model where authentication is one aspect between microcontrollers and applications. The authors in [30] by creating an architecture that provides data only to trusted third-party applications by using the north interface along with the implementation of the NSS digital signature and the NTRU encryption method. Similarly, [31] he proposes a framework consisting of two main modules: a detection engine responsible for permissions, which determines the legitimacy of application permissions, and a logging authorization engine, which limits eavesdropping attacks and manipulation, where it uses the NTRU method to perform both logging and authorization of the request [10].

4.2 Security of East/West Bound Interface and Control Plane

The controller or network operating system that manages the requests made from the data plane logically controls SDN networks. Structure information, application resource creation, statistics, and much more are all contained in the controller. There are now a large number of controllers, maybe more than thirty, each with its own programming languages and interface; many are open source, while others are proprietary. The architecture of controllers can be characterized as centralized or distributed [32, 33].

According to the literature, there is a sub-classification of distributed systems that can assume either flat or hierarchical shapes. The tasks of each controller within the organization are the basis for this sub-classification [33]. As a result, all controllers in a flat distributed architecture have the same applications and responsibilities. Hierarchical distributed architectures contain a powerful controller that can handle applications within the network and fewer applications and responsibilities with more controllers [32]. Despite this, there is no strong significance as flat and hierarchical distributed architectures are one and the same for security issues and hence controllers are referred to as central or distributed. Modules are used in applications where throughput is low as they rely on a single controller to manage the network. As a result, the use of a unique controller may cause a single point of failure that damages the network, affecting its performance and leading to bottlenecks for incoming requests, so this architecture may be vulnerable to SDN attacks [10].

On the other hand, there are more controllers in multi-domain or heterogeneous networks [34] used by distributed designs as they are mainly used in large-scale deployments by telecom operators for different functions within the network. Distributed controllers try to configure the network in terms of scalability, throughput, and flexibility, as it is very resilient in the presence of DoS attacks, as the controller uses fault tolerance technology to manage this function. He cites Hyperflow [35] which uses the oath-bearing property of WheelFS's system. In distributed architectures, the communication between the controllers is achieved through the east/west interface, which is not uniform like the north interface, so each controller proposes an interface. The eastern and western interfaces, respectively, provide communication between the controllers in the various administrative fields, and the other interfaces link traditional networks with SDN networks [36] [10].

There are also solutions that use methods to solve the problem of SDN security of pre-defined networks such as methods based on statistics and methods based on machine/deep learning [38]. In the context of this, there are some helpful proposals to solve this problem: In [39] researchers proposed IDS called Eunoia, which uses machine learning and consists of three subsystems: data processing, which is responsible for getting rid of unimportant data through statistics and traffic analysis after that. The extracted data is sent to the second sub-component: predictive data modeling, which uses a random forest algorithm to detect attackers at the same time. The third component is decision-making and response systems subcomponents, which work on inaccurate data to achieve greater accuracy of the extracted data.

The authors in [40] proposed a framework for anomaly detection using deep learning. The proposal consists of two modules, the flaw detection module, which is responsible for the security aspect and focuses on many attack vectors such as spoofing and malware. This unit is based on the so-called support vector machine (SVM) algorithm and restricted Boltzmann machine (RBM) whereby an anomaly report is generated by classifying and summarizing the characteristics of the flows sent between the controller and network parts; based on this report, the controller can work on forwarding while disregarding packets and communication. The second module is responsible for the quality of data delivery to provide Quality of Experience (QoE).

The study in [41] focuses its work on application-type attacks such as cross-site scripting, botnets, and port scanning. It is a framework for detecting eavesdropping based on hybrid learning techniques and the application of long-term memory and long short-term memory (LSTM) which is useful to avoid the vanishing gradient problem found in a large data sequence set. This proposal also uses a convolutional neural network (CNN) to extract features from the raw data.

In [42] the authors propose an IDS using deep learning. This solution consists of three modules: 1) Stream collector which is responsible for collecting all sensitive packet-in messages. 2) Anomaly detector, in which the anomaly is detected using a gated recurrent unit recurrent neural network (GRU-RNN). 3) Anomaly mitigator in which it is decided whether the traffic is to be ignored or analyzed in depth.

4.3 Security of SDN Data Plane

The flow rules are inserted into the flow tables of OpenFlow switches by means of the OpenFlow controller in OpenFlow networks. These rules can be created before a new host delivers packets or after a new host sends the first packet. As shown in Figure 3, the switch has a limited number of flow tables where the flow rules are installed according to the controller width of the network. Where the most necessary priority is to identify and separate the normal flow rules from the anomalous flow rules, followed by the important priority, which is the ability of the transformer to maintain an appropriate amount of flow inputs. In OpenFlow, it is important that the switch stores temporary streams and waits for the controller to issue stream rules. The data plane has limited resources to store unwanted temporary packets (TCP/UDP) and it is vulnerable to attack, especially saturation attacks [43].

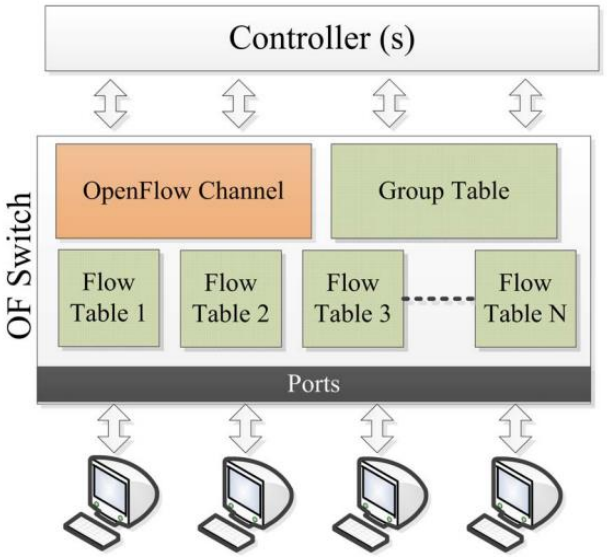

Figure 3. OpenFlow switch architecture [43]

In SDNs, the data plane is affected by the security of the control plane. Therefore, in the event of an attack on the control plane, the entire network, which includes many data plane nodes, is compromised. In partitioned network architectures such as SDNs, a control-level attack may fail or be disconnected, so the redirection information does not reach the switch and therefore is attacked at that time. When the control plane is separated from the data plane, an attacker can manipulate OpenFlow's rules causing more attacks in SDNs such as man-in-the-middle attacks and black-hole attacks. The TLS protocol has a higher technical barrier for operators due to its complex configuration, which includes creating a site-wide certificate, signing it with the private key, creating control and switching certificates, and installing the correct keys and certificates on devices. As a result, several suppliers have decided to leave TLS functionality out of their OpenFlow switches. Complex configuration and selective use of TLS can cause the control channel to be attacked. One of the most common types of attacks in OpenFlow networks is the man-in-the-middle attack due to the high connectivity and lack of authentication in the plaintext OpenFlow TCP control channel. Then the attacker can be able to access any downstream switches and perform a lot of eavesdropping attacks [43].

V. Conclusions and Future Scope

The SDN weakness point is the SDN controller as it may be a single point of attack that compromises the whole network. Therefore, special considerations shall be applied for securing the SDN controller. In this research, we presented an overview of security threats for Software Defined Networking (SDN) and we detailed countermeasures to defend SDN from such security threats. Our future plan is to develop security tools for protecting the SDN controller against various security threats, including Denial of Service (DoS) attacks and Distributed Denial of Service (DDoS) attacks.

REFERENCES

[1] Ali, A. 2001.Macroeconomic variables as common pervasive risk factors and the empirical content of the Arbitrage Pricing Theory. Journal of Empirical finance, 5(3): 221–240.

[2] Basu, S. 1997. The Investment Performance of Common Stocks in Relation to their Price to Earnings Ratio: A Test of the Efficient Markets Hypothesis. Journal of Finance, 33(3): 663-682.

[3] Bhatti, U. and Hanif. M. 2010. Validity of Capital Assets Pricing Model.Evidence from KSE-Pakistan.European Journal of Economics, Finance and Administrative Science, 3 (20).

[1] Bannour, Fetia and Souihi, Sami and Mellouk, Abdelhamid, "Distributed SDN control: Survey, taxonomy, and challenges," IEEE Communications Surveys \& Tutorials, vol. 20, no. 1, pp. 333—354, 2017.

[2] A. Abdou, P. C. van Oorschot and T. Wan, "Comparative Analysis of Control Plane Security of SDN and Conventional Networks," IEEE Communications, vol. 20, pp. 3542-3559, 2018.

[3] Gadallah, Waheed G and Omar, Nagwa M and Ibrahim, Hosny M, "Machine Learning-based Distributed Denial of Service Attacks Detection Technique using New Features in Software-defined Networks," International Journal of Computer Network and Information Security (IJCNIS), vol. 13, no. 3, 15—27, 2021.

[4] H. Zhang, Z. Cai , Q. Liu , Q. Xiao, Y. Li, and C. F. Cheang , "A Survey on Security-Aware Measurement in SDN," Security and Communication Networks, 2018.

[5] Singh, Sarika and Prakash, Shiva, "A Survey on Software Defined Network based on Architecture, Issues and Challenges," IEEE 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC), pp. 568—573, 2019.

[6] C.E. Rothenberg et al. "Software defined networking: A comprehensive survey" proceedings of the IEEE, volume. 103, num. 1, pp. 14-76, 2015.

[7] H. Song, "Protocol-oblivious forwarding unleash the power of SDN through a future proof forwarding plane" In processing second ACM SIGCOMM workshop hot topics software defined network. pp. 127- 132, 2013.

[8] Burkhalter, Lukas and Lycklama, Hidde and Viand, Alexander and K{\"u}chler, Nicolas and Hithnawi, Anwar, "Rofl: Attestable robustness for secure federated learning," arXiv preprint arXiv:2107.03311, 2021.

[9] Xia, Wenfeng and Wen, Yonggang and Foh, Chuan Heng and Niyato, Dusit and Xie, Haiyong, "A survey on software-defined networking," IEEE Communications Surveys \& Tutorials, vol. 17, no. 1, pp. 27—51, 2014.

[10] Jim{\'e}nez, Mar{\'\i}a B and Fern{\'a}ndez, David and Rivadeneira, Jorge Eduardo and Bellido, Luis and C{\'a}rdenas, Andr{\'e}s, "A Survey of the Main Security Issues and Solutions for the SDN Architecture," IEEE Access, vol. 9, pp. 122016—122038, 2021.

[11] P. V. Tijare and D. Vasudevan, ''The northbound APIs of software de ned networks,'' Int. J. Eng. Sci. Res. Technol., vol. 5, pp. 501 513, Jan. 2019. [Online]. Available: http://www.ijesrt.com

[12] S. Y. Zhu, S. Scott-Hayward, L. Jacquin, and R. Hill, Guide to SecurITY SDN 78 NFV: Challenges, Opportunities, Application. Cham, Switzerland: Springer, 2017.

[13] P. Ahmad, S. Jacob, and R. Khondoker, ''Security analysis of SDN applications for big data,'' in SDN and NFV Security (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 39 55.

[14] D. Artmann and R. Khondoker, ''Security analysis of SDN WiFi applications,'' in SDN and NFV Security (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 57 71.

[15] M. Bräuning and R. Khondoker, ''Analysis of SDN applications for smart grid infrastructures,'' in SDN and NFV Security (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 99 110.

[16] A. Chikhale and R. Khondoker, ''Security analysis of SDN cloud applications,'' in SDN and NFV Security (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 19 38.

[17] R. Jain and R. Khondoker, ''Security analysis of SDN WAN applications B4 and IWAN,'' in SDN and NFV Security (Lecture Notes in Networks and Systems), vol. 30. Cham, Switzerland: Springer, 2018, pp. 111 127.

[18] C. Lee, C. Yoon, S. Shin, and S. K. Cha, ''INDAGO: A new framework for detecting malicious SDN applications,'' in Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP), Sep. 2018, pp. 220 230.

[19] Y. Li, Z. Wang, J. Yao, X. Yin, X. Shi, J. Wu, and H. Zhang, ''MSAID: Automated detection of interference in multiple SDN applications,'' Comput. Netw., vol. 153, pp. 49 62, Apr. 2019.

[20] T. Hu, P. Yi, Y. Hu, J. Lan, Z. Zhang, and Z. Li, ''SAIDE: Ef cient application interference detection and elimination in SDN,'' Comput. Netw., vol. 183, Dec. 2020, Art. no. 107619.

[21] M. Latah and L. Toker, ''Load and stress testing for SDN's northbound API,'' Social Netw. Appl. Sci., vol. 2, no. 1, Dec. 2019, Art. no. 122, doi: 10.1007/s42452-019-1917-y.

[22] B. E. Ujcich, S. Jero, A. Edmundson, Q. Wang, R. Skowyra, J. Landry, A. Bates, W. H. Sanders, C. Nita-Rotaru, and H. Okhravi, ''Cross-app poisoning in software-de ned networking,'' in Proc. ACM Conf. Comput. Commun. Secur. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 648 663. [Online]. Available: https://dl.acm.org/doi/10.1145/3243734.3243759

[23] R. Chang, Z. Lin, Y. Sun, and J. Xu, ''MD-UCON: A multi-domain access control model for SDN northbound interfaces,'' J. Phys., Conf. Ser., vol. 1187, no. 3. 2019, Art. no. 32091.

[24] H. D. Hoang, P. T. Duy, and V.-H. Pham, ''A security-enhanced monitoring system for northbound interface in SDN using blockchain,'' in Proc. 10th Int. Symp. Inf. Commun. Technol., New York, NY, USA, 2019, pp. 197 204.

[25] Y. Tseng, M. Pattaranantakul, R. He, Z. Zhang, and F. Nait-Abdesselam, ''Controller DAC: Securing SDN controller with dynamic access control,'' in Proc. IEEE Int. Conf. Commun. (ICC), May 2017, pp. 1 6.

[26] B. Toshniwal, K. D. Joshi, P. Shrivastava, and K. Kataoka, ''BEAM: Behavior-based access control mechanism for SDN applications,'' in Proc. 28th Int. Conf. Comput. Commun. Netw. (ICCCN), Jul. 2019, pp. 1 2.

[27] Y. Tseng, F. Nait-Abdesselam, and A. Khokhar, ''SENAD: Securing network application deployment in software de ned networks,'' in Proc. IEEE Int. Conf. Commun. (ICC), May 2018, pp. 1 6.

[28] H. Cui, Z. Chen, L. Yu, K. Xie, and Z. Xia, ''Authentication mechanism for network applications in SDN environments,'' in Proc. 20th Int. Symp. Wireless Pers. Multimedia Commun. (WPMC), Dec. 2017, pp. 1 5.

[29] G. Kim, J. An, and K. Kim, ''A study on authentication mechanism in SEaaS for SDN,'' in Proc. 11th Int. Conf. Ubiquitous Inf. Manage. Commun., New York, NY, USA, Jan. 2017, pp. 1 6. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3022227.3022277

[30] S. B. H. Natanzi and M. R. Majma, ''Secure northbound interface for SDN applications with NTRU public key infrastructure,'' in Proc. IEEE 4th Int. Conf. Knowl.-Based Eng. Innov. (KBEI), Dec. 2017, pp. 452 458.

[31] T. Hu, Z. Zhang, P. Yi, D. Liang, Z. Li, Q. Ren, Y. Hu, and J. Lan, ''SEAPP: A secure application management framework based on REST API access control in SDN-enabled cloud environment,'' J. Parallel Distrib. Comput., vol. 147, pp. 108 123, Jan. 2021.

[32] M. Karakus and A. Durresi, ''A survey: Control plane scalability issues and approaches in software-de ned networking (SDN),'' Comput. Netw., vol. 112, pp. 279 293, Jan. 2017.

[33] L. Zhu, M. Monjurul Karim, K. Sharif, F. Li, X. Du, and M. Guizani, ''SDN controllers: Benchmarking performance evaluation,'' 2019, arXiv:1902.04491. [Online]. Available: http://arxiv.org/abs/1902.04491

[34] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, ''A survey on software de ned networking with multiple controllers,'' J. Netw. Comput. Appl., vol. 103, pp. 101 118, Feb. 2018.

[35] A. Tootoonchian and Y. Ganjali, ''HyperFlow: A distributed control plane for open ow,'' in Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw., vol. 3, 2010, pp. 1 8.

[36] Z. Latif, K. Sharif, F. Li, M. Monjurul Karim, and Y. Wang, ''A comprehensive survey of interface protocols for software de ned networks,'' 2019, arXiv:1902.07913. [Online]. Available: http://arxiv.org/ abs/1902.07913

[37] S. Khan, A. Gani, A. W. Abdul Wahab, M. Guizani, and M. K. Khan, ''Topology discovery in software de ned networks: Threats, taxonomy, and state-of-the-art,'' IEEE Commun. Surveys Tuts., vol. 19, no. 1, pp. 303 324, 1st Quart., 2017.

[38] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, ''Survey of intrusion detection systems: Techniques, datasets and challenges,'' Cybersecurity, vol. 2, no. 1, pp. 1 22, Dec. 2019. [Online]. Available: https://link.springer.com/articles/10.1186/s42400-019-0038-7.

[39] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, ''Machine-learning based threat-aware system in software de ned networks,'' in Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN), Jul. 2017, pp. 1 9.

[40] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, ''Hybrid deep-learning-based anomaly detection scheme for suspicious ow detection in SDN: A social multimedia perspective,'' IEEE Trans. Multimedia, vol. 21, no. 3, pp. 566 578, Mar. 2019.

[41] J. Malik, A. Akhunzada, I. Bibi, M. Imran, A. Musaddiq, and S. W. Kim, ''Hybrid deep learning: An ef cient reconnaissance and surveillance detection mechanism in SDN,'' IEEE Access, vol. 8, pp. 134695 134706, 2020.

[42] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, and M. Ghogho, ''Intrusion detection in sdn-based networks: Deep recurrent neural network approach,'' in Deep Learning Applications for Cyber Security (Advanced Sciences and Technologies for Security Applications). Cham, Switzerland: Springer, 2019, pp. 175 195. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-13057-2_8

[43] Ahmad, Ijaz and Namal, Suneth and Ylianttila, Mika and Gurtov, Andrei, "Security in software defined networks: A survey," IEEE Communications Surveys \& Tutorials, vol. 17, no. 4, pp. 2317—2346, 2015.