International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Scalable Framework to Detect, Analyze, and Prevent Security Vulner abilities in Enterprise Software-Defined Networks

***Bhumika Sapkota*[1] *and Akalanka B. Mailewa*[2, *]**

[1]Department of Information Systems, St. Cloud State University, St. Cloud, Minnesota, USA
bsapkota2@go.stcloudstate.edu

[2] Department of Computer Science and Information Technology, St.Cloud State University,St. Cloud, Minnesota, USA
amailewa@stcloudstate.edu

Doi: https://doi.org/10.55248/gengpi.2022.3.2.1

A B S T R A C T

Software Defined Networking (SDN) is a rapidly growing technology that is enabling innovation on how network systems are designed and managed. Like any other technology, SDN is susceptible to numerous security threats. The separation of planes and centralized control topology of SDN makes it vulnerable to myriad of attacks. There has been rapid implementation of SDN in variety of networks and this is only growing with time. There are only handful of resources for enterprise networks that are actively transitioning their networks to SDN and require security specification.This paper proposes a security framework that integrates basic security requirements for SDN. First, the security vulnerabilities are identified by assessing the SDN topology. This provides understanding of the issue which is utilized to implement security mechanisms that help mitigate the vulnerabilities. The framework utilizes open-source tools and techniques and integrates well-known security mechanisms. Throughout the paper, the proposed framework is implemented and tested for its success in satisfying some of the basic and crucial security requirements. In doing so, this research offers viable security mechanisms for enterprise networks that are looking for performance and cost-effective security solutions.

Keywords:Security, Vulnerabilities, Attacks, SDN, OpenFlow, Ryu-controller

1. Introduction

Software-defined networking is a fast-paced technology that is being adapted in networks all over the world. It has changed the traditional way of network management by shifting the functions of different network components for programmability and added flexibility. RFC 7426 defines SDN as a programmable network approach that supports the separation of control and forwarding planes via standardized interfaces [1]. This separation of planes in a network makes network switches and routers a simple forwarding device without the backlog of decision-making. The central logic implemented in the controller is then responsible for network management and policy implementation [2].

The functionality of SDN has skyrocketed its deployment rates. Recent research has shown that the SDN market is expected to grow from USD 13.7 billion in 2020 to USD 32.7 billion by 2025 [3]. The exponential growth is only likely to rise further, with SDN being widely used in up-and-coming technologies like cloud computing and virtualization. Thus, the security of these networks is a crucial topic that must be acknowledged along with its growth span. A well-defined security framework can leverage the flexibility of SDN architecture and factor in the necessities of enterprise networks. Therefore, this paper focuses on developing a framework that defines security mechanisms for enterprise network architecture by considering factors like performance, cost, and flexibility. This paper aims to make it easier for enterprises to integrate security mechanisms in SDN environments seamlessly.

* *Corresponding Author: Akalanka B. Mailewa*
E-mail address: amailewa@stcloudstate.edu

1.1. Motivation

According to Gartner's Magic Quadrant for WAN Edge infrastructure report, only 35% of SD-WAN networks have implemented secure architecture [4]. For a technology with a global market size of USD 13.7 billion in 2020, the security implementation percentage is concerning. Compared to the SDN deployments in enterprise networks and data centers, the available security specifications, and guidelines for SDN are severely lacking. One of the primary motivations for this research is to bridge the gap between the necessity and availability of a well-tested security framework for enterprise networks.

Another motivation for this research is the feature of SDN itself. SDN offers excellent flexibility and functionality for managing networks. The primary stronghold of SDN, i.e., separation of control and data plane and the global view of the network, can be leveraged to integrate several security mechanisms. However, the documentation available for reference and guidelines is sparse. Open Networking forum, one of the most significant contributors for SDN-related research and documentation, has made some technical specifications and standards available. But these documents lack in context of providing specific security implementation and analyzing its impact. Therefore, this paper aims to develop a framework that assures authentication, authorization, and availability by leveraging SDN architecture.

1.2. The Problem Description

The lack of proper guidelines and security framework for enterprises deploying SDN in their environment is a concerning issue. While SDN has changed the attack surface of traditional networks, it has introduced a new realm that can be exploited [5]. Despite the surge in enterprises adopting SDN, there is a lack of security specifications, or a security framework adapted for security implementation. Lack of specification also means that many enterprises overlook security implementation because of added overhead and higher technical barriers [6].

1.3. Objectives

The main objective of this paper is to propose a security framework that utilizes well-known security mechanisms and leverages the features of SDN for easier management. It focuses on:

1. Studying and identifying major security issues within SDN deployed enterprise environment.
2. Proposing and implementing security mechanisms to mitigate risks and vulnerabilities.
3. Proposing a security framework for enhancing security in enterprise infrastructure.

1.4. Research Questions

- RQ1: How can security prerequisites be identified in Software-Defined Networks (SDN)?
- RQ2: What vulnerabilities and attack surfaces are associated with Software Defined Networks (SDN) topology?
- RQ3: How can the security of SDN be improved? Can SDN architecture be leveraged to enhance the security of the SDN?

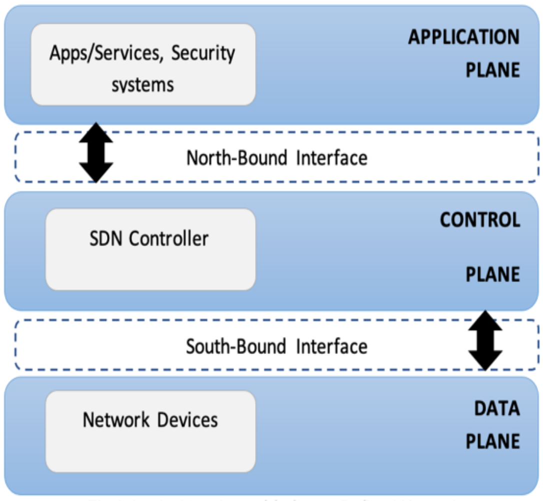
Software-Defined Networking technology enables virtualization in an everyday network by separating planes and integrating programmability in the centralized controller system. The programmability of controllers offers an abstraction from lower-level infrastructure, making it possible to simplify policy-based routing and management instead of using proprietary protocols and vendor-based configuration sets [7]. In addition, the topology of SDN, which is different from traditional networks, enables the segregation of network control and packet forwarding in SDN.A typical SDN architecture can be segregated into three planes: Application plane, Control Plane, and Data Plane. Each plane communicates with the other through a specified interface. Each component of SDN architecture is explained in detail below:

1. **Application Plane:**This plane includes applications and services, mainly comprised of end-user applications that consume network services [8]. Services residing in the application plane include network topology discovery, network provisioning, and security systems like IDS/IPS and monitoring services. These services work in conjunction with the control plane via the northbound interface. This communication contains parameters like delay, throughput, and availability descriptors, providing applications a more comprehensive view of the network [9].

2. **Control Plane:** The control plane is the core of SDN architecture. It is responsible for controlling packet forwarding, switching decisions, and configuring the data plane using a southbound interface. Being the powerhouse of SDN, its functionalities include but are not limited to topology discovery and maintenance, route selection, and policy control. SDN controller answers individual application requests and objectives such as traffic prioritizing, access control, bandwidth management and relays it to the data plane components [9]. Admins can write and rewrite rules for how network traffic, data packets, and frames traverse the network and how the network infrastructure in this layer handles them. This layer also enables support for business applications like SLAs, QoS, and Policy Management [10]

3. **Data Plane:**The Data plane includes network devices and physical resources such as switches, routers, and hubs which forward packets based on central policies [8]. It is also termed the infrastructure layer because of the physical and virtual devices it includes. Responsibilities of data plane include packet forwarding, dropping, managing operation state of network devices, etc.

4. **Southbound Interface:**The southbound interface is the communication interface between the control plane and data plane, which allows the controller to interact with the data plane elements. The control plane communicates network policies, flow tables, and network updates to and



from the data plane entities via the southbound interface [11]. SBI supports and uses different protocols for this communication. However, OpenFlow protocol is the standard communication protocol for the southbound interface. Communication in this interface happens over TCP.

Figure 1. SDN logical topology

5. **Northbound Interface:**The northbound interface is the interface used by applications and controllers to interact with each other. This interface enables the programmability of the controllers by exposing the network abstraction data model and other functionalities within the controllers [12]. Unlike the southbound interface, there is no standardized protocol for northbound communication yet. Thus,the communication highly relies on vendor-proprietary methods.

6. **OpenFlow:** It is a communication protocol that allows direct access and manipulation of the forwarding plane of network devices, both physical and virtual [13]. It is a standard API used by SDN for communication among different planes.

2. Review of Literature

A recent study [14] proposes a secure framework for SDN-based Edge computing in IoT-enabled healthcare systems. The framework includes a lightweight authentication scheme to authenticate the IoT devices in the network. The authentication scheme authenticates IoT devices by identifying them based on the operating frequency bands, i.e., access frequencies or time slots. Upon authentication, the devices collect data from the end-user and send them to an Edge server with a central controller for storage, processing, and analysis. The proposed framework helps improve security in devices with less computational capacity by introducing a secure authentication mechanism.

Another study [15] proposes a distributed SDN framework-Tennison for scalable network security. The framework presents a distributed implementation that includes multiple control instances, tunneling for efficient attack detection and mitigation, and multi-level monitoring. The multiple levels of monitoring include lightweight monitoring that can handle high volume of data flow. This framework aims in providing scalable and transparent security. The framework aids security by optimizing network monitoring and protection through appropriate switch/controller assignment and monitoring rule placement. This study has future potential for scaling by automating the processes.

Cisco’s Zero Trust, Zero Touch [16] is another recent innovation that integrates a layered security approach for network security. This approach incorporates visibility into the control, data, orchestration, and management planes in Software-defined networks with a goal of faster threat detection/remediation. This SDN approach set forward by Cisco bases on defense-in-depth techniques. It includes multiple layers of protection automated by leveraging SDN’s programmability. This framework provides a holistic approach to securing the entire network by taking advantage of programmability and visibility.

Another study [17] proposes a framework for SDN deployment in data centers. The framework provides an adaptive self-defending network that can detect threats by analyzing network traffic and searching for anomalous patterns. It integrates series of security monitoring such as Intrusion Detection System and Intrusion Prevention System backed by a Syslog service. The framework’s control unit contains the security agent unit and Syslog server, and it integrates with the SDN controller. One of the significant advantages of this framework is that it supports out-of-band management for network traffic so that it does not affect network latency for services. Furthermore, the independent design of security agent from the SDN controller adds on to the scalability of this framework.

Another study on controller security proposes [18] a trust establishment framework between SDN controller and applications. The trust framework uses a direct trust establishment mechanism based on observing activities performed by SDN applications when requesting network resources. Trust establishment is complete between the controller and applications upon the completion of direct trust calculation. This framework provides security to the controller from malicious and rogue devices and secures SDN from many threats.

Karmakar et. Al, propose a security architecture [19] for SDN infrastructure that uses a policy-based approach to coordinate different security mechanisms. The dynamic policy framework presented in this architecture enables the detection and mitigation of security attacks. It leverages feedback between various security components and mechanisms. The policies put constraints over users, flows, services, IP, MAC, and port which allows for controlling and managing the flow communication. The proposed framework also integrates a trust management framework to evaluate the trustworthiness of the OpenFlow switch at run-time. In addition, it includes a signature-based intrusion detection system to detect intrusions. A key management mechanism is in place to maintain secure communication between the central controller and the OpenFlow switches.
Hussein et. Al,. propose an SDN security plane [20] for resilient security services in software-defined networks. The proposal presents an additional plane dedicated to security. The security plane is responsible for forwarding data packets between the data plane and the control plane. A security module is present before the controller, responsible for collecting and analyzing all data traffic coming from all agents. This module helps detect abnormal events and trigger alarms to the controller. The framework includes DDoS prevention by including a threshold to compare the collected traffic.

Shi Et. Al,. propose a security architecture [21] that aims to provide defense ability to the infrastructure corresponding to each plane. The proposal includes an access control strategy based on attribute-based encryption. The encryption uses a tree structure to achieve fine-grained SDN access control. The proposed mechanism comprises four components: a trusted server to manage identity, the attributed center, the access subject, and the access object.

Pandya Et. Al,. propose a framework [22] for securing SDN southbound communication to improve the security posture. The southbound APIs are the connecting channel between the control plane and the data plane. The security of the southbound interface ensures the integrity and confidentiality of the communication between the control and forwarding elements. The paper suggests a framework that leverages security systems similar to middleboxes and a key distribution center to digitally signed messages for addressing data integrity and authorization. The middle-box solution ensures that only authorized devices have access to the network is accessed, and the Key Distribution Center (KDC) provides authentication.

Sallam et. Al. propose Software Defined Perimeter (SDP) [23] to manage the connections in software-defined networks and inherently build a secure environment. Some of the primary concerns in SDN are authentication, authorization, access control, privacy, and integrity. Integrating security systems within SDN helps with tackling security issues. A typical SDP consists of three main components: Initiating host, accepting host, and the controller. These components create secure perimeters among legitimate clients and network services. The proposed framework is tested in a virtual testbed for effectiveness in integrating and enhancing security.

2.1. Current Issues in the Research

In software-defined networks, the control plane oversees decision-making process based on the global view of the network. The flow rule determines the flow of data packets. An attacker can bypass security devices that lack notable security measures implementation. Also, because of SDN's architecture, it is relatively easy for the attacker to obtain a global view of the network and learn about the current security posture of the network [24].

A study done on vulnerabilities related to flow rule observed that attackers could bypass rules or policies by running malicious applications that can construct overlapping and conflicting flow rules. This attack leverages the fact that controllers cannot distinguish between pre-bound and newly issued rules [25]. It presents threats of network misconfiguration, malfunction, denial of service, and other threats if leveraged in a certain way by an attacker.
Another study done in the SDN network showed how an attacker could exploit the intrinsic property of SDN architecture. If the data plane comes across packets with no handling information during data flow, the data plane reaches out to the control plane for flow information. A spoofed device can continuously send out such packets to the control plane to exhaust the communication channel and the controller itself. It also gives an insight to an attacker if the network uses OpenFlow switches, which can be used to generate specially crafted flow requests towards the controller [26]. An attacker can use this kind of attack to wear out the controller prompting a Denial-of-Service attack.

Another study found a threat of link fabrication attack in SDN connection protocol. SDN uses Link Layer Discovery Protocol (LLDP) for internal link discovery service. However, it cannot verify the legitimacy of LLDP packets or the patch of the packets during the transmission. Attackers can exploit this overlook on security to execute link fabrication attacks. Link fabrication attacks are used by adversaries to inject fake links in the network through which they can control the traffic which traverses the malicious link. These attacks can take any form, from forgery to relay [27] [28].

OpenFlow is one of the most common technologies used by SDN networks. There are many studies on the workings and security of the protocol and design of OpenFlow. While OpenFlow switch specification describes using TLS as a communication protocol, the feature is optional. Furthermore, there is no specification on any other authentication mechanism for communication between the control and the data plane [29]. This lack of authentication mechanism is a significant overlook on security. Studies have shown that attackers can exploit this vulnerability for differe nt man-in-the-middle attacks and DoS attacks. It paves an easy path for attackers to manipulate and control the central management system by hijacking a communication channel.

There are several similar studies done on SDN networks and their architecture. Most studies have noted that SDN networks' prime security issues are a bottleneck between data plane and control plane, lack of secure communication channel, and issues related to central control. These issues lead to potential risks of DoS attacks, spoofing attacks, and Man-in-the-middle attacks. The segregation of control and data plane in SDN networks make up for different attack surfaces. The data layer, application layer, controller, OpenFlow switches, communication protocol, and traffic flow are the attack surface that an attacker can exploit in many ways [30]. The analysis of SDN networks suggests that the very features that an attacker can exploit can be used to secure the network.

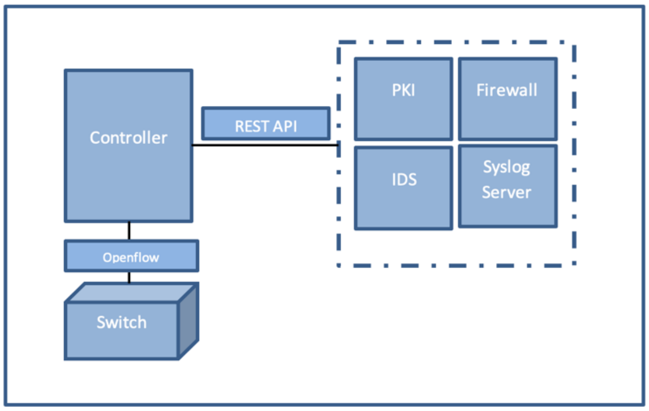


Figure 2. Proposed Security framework

3. Methodology

3.1 The implementation and test phase are broken down into two stages. Firstly, the test SDN environment is setup and assessed for security vulnerabilities. Then, the proposed security framework is implemented. The network is assessed for vulnerabilities post implementation and the collected data is analysed.

3.2. Hardware and Software Environment

- Virtual Environment: VirtualBox 6.1.16
- Operating System: Ubuntu 18.04 LTS desktop
- RAM: 16.0 GB
- Software Environment: Mininet 2.2.2

3.3. Design of the study

The proposed framework includes a security unit that integrates different security mechanisms including authentication, authorization, and monitoring. The framework is integrated into the SDN topology through the northbound interface. The framework provides flexibility to be tailored as per individual requirements, i.e., it supports integrating additional security mechanisms via the northbound interface. OpenFlow is configured for communication among network entities. OpenFlow supports TLS v1.2 and above and PKI can be used to establish certificate validity chains [31]. The framework uses ovs-pki script to create an initial PKI structure [32]. Then, a private key and certificate are generated for both switch and controller. Next, the openvswitch daemon is configured to use CA files. Finally, the Ryu controller is run with the CA files. This process sets up a TLS connection for Controller-Switch communication.While policy-based access control mechanisms are widely used, the ONF openflow specification [33] for SDN does not mandate or suggest the use of any specific authorization mechanisms. Therefore, for authorization purposes, the framework integrates rule-based access control mechanism by using a stateful firewall. Firewalls are commonly used in systems and networks for security purposes. They allow flexibility to define rules as needed for a particular scenario [34]. Moreover, since stateful firewalls filter TCP packets based on the matching rule and state of the three-way handshake, it is ideal for access control purposes.A stateful firewall application written in Python is taken and integrated with the Ryu controller for this framework. Upon receipt of a packet, the switch will forward only the first few packets of the new flow to the controller. Then, according to the policies, the firewall application will prepare flow table entries and push the flow table to openvswitch. This process ensures that notall packets need to be individually handled by the controller and offloads excessive controller use [35]. For monitoring and IDS/IPS roles, the framework integrates Snort. Snort [36] is an open-source intrusion detection and prevention system that works as a packet sniffer, packet logger, and intrusion prevention system. Snort allows the creation and implementation of a ruleset according to requirements. Snort can also be deployed inline production to block/flag traffic and generate an alert. In the proposed framework, Snort integrates with the controller via the northbound interface for granular detection and blocking. The Ryu controller plays a vital role in managing snort rules and openvswitch entries [37]. Another vital part of the network and system security is logging. Syslog (One identity llc, 2021) allows real-time log collection and processing. Its functions also include log parsing, classification, and correlation. It also gives the added flexibility to be integrated with various network applications to efficiently monitor and inventory logs [38]. Syslog server can be used in tandem with Snort to send logs and alerts for auditing and accounting purposes. Logs are also highly essential in identifying anomalies, setting a benchmark, and for investigative purposes.In the first phase of this study, Nmap is used for network discovery and scanning nodes for open ports and services. Several probes for host discovery, discovering services and open ports on nodes, system description, and uptime are done using Nmap. Similarly, Wireshark captures packets in the southbound interface between the OpenFlow switch and the controller and the connected hosts. In this test topology, the SDN controller is local, and thus the controller and OpenFlow switch talk over the local interface. Therefore, we capture the packet of the loopback 0 interfaces. Communication between OpenFlow switch and hosts is captured through eth1 and eth2 interfaces. For generating traffic, the pingall command is executed in the Mininet setup. When this command is executed, H1 sends an ICMP packet to S1, which queries Controller C0 for a route to H2. The controller sends flow information to S1, which S1 uses to forward the packet to S2. Once this flow information is obtained, the packets flow between H1-S1-H2 directly until the flow information expires from the buffer. This process triggers a series of communication in the topology. For the second phase tests, packet captures and ICMP probe tests are done to verify the effectiveness of security mechanisms.

3.4. Performance Metrics

In order to test the effectiveness of the framework, series of tests are conducted pre and post framework implementation. The results of the tests help determine how successful the framework is in assuring security. The tests are conducted against number of crucial security requirements for SDN environments. For this specific study, we have referenced the security requirements from the Open Networking Foundation's security requirements for SDN [39]. These requirements include but are not limited to authentication, authorization, and availability.

4. Results and Discussion

In this section, we analyze the results of the security assessment pre and post implementation of the framework. In order to evaluate the efficiency of the framework, we run tests against the SDN security requirements defined by the Open Networking Foundation. Firstly, security requirements are identified by assessing the security of the network through different tests. The results are summarized below.

1. No authentication between controller and switch

   Category: Controller Vulnerability
   Root Cause: Authentication mechanism is not set
   Prevention Technique: Certificate and shared keys are common method for identity verification. TLSv.1.2 implementation with PKI is suggested.
   Security Requirement: Authentication on SDN controller interface
2. User Authentication Not Set

   Category: Controller Vulnerability

   Root Cause: No user authentication mechanism

   Prevention Technique: Use of one or more authentication technique using a unique identity.

   Security Requirement: User Authentication
3. Data in Transit is unencrypted

   Category: Controller Vulnerability

   Root Cause: Encryption mechanism not set

   Prevention Technique: Openflow supports security protocol TLS for securing communication in transit [40]. The newer version of TLS i.e., v1.2 or higher can be used to secure communication between controller and the entities.

   Security Requirement: Data-in-transit encryption
4. Unnecessary ports are open

   Category: Controller Vulnerabilities

   Root Cause: Network/systems are not hardened according to best common practices

   Prevention Technique: Assess network components for identifying unnecessary ports and services identification. Block open ports that are not actively used.

   Security Requirement: Close unnecessary ports/services
5. Log function is not defined

   Category: Network/Controller vulnerability

   Root Cause: Lack of Log function

   Prevention Technique: Implement logging solutions such as Syslog server. Also, parameters which helps in incident investigation such as username, timestamp, performed action etc. should be captured in the logs.

   Security Requirement: Log function
6. No IP check function in the controller

   Category: Controller vulnerability

   Root Cause: IP whitelisting not configured

Prevention Technique: Identify the list of trusted IP address and configure IP whitelisting

Security Requirement: IP check

7. No privilege control of application

   Category: Controller Vulnerabilities

   Root Cause: Application privilege not defined

   Prevention Technique: Review and allocate application privilege based on the least privilege principle.

   Security Requirement: Application privilege control.

8. No DDoS prevention mechanisms

   Category: Controller vulnerabilities

   Root Cause: Lack of monitoring and pre-set threshold for network activities

   Prevention Technique: Monitoring and threshold configuration

   Security Requirement: Anti-DoS mechanisms

9. ARP is broadcast network wide

   Category: Controller Vulnerability

   Root Cause: Authentication/Authorization of device is not present

   Prevention Technique: Authorization of devices can be configured to ensure that only authorized devices respond to ARP broadcast. IP whitelisting in controller also helps eliminate risks of rogue device attempting ARP spoofing.

   Security Requirement: Authenticate devices

10. OFDP protocol is used for topology discovery

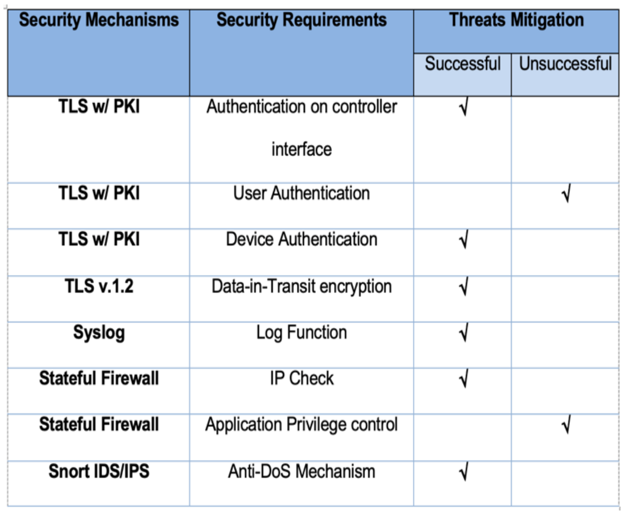
    Category: Controller Vulnerabilities

    Root Cause: No authentication of LLDP messages

    Prevention Technique: Device authentication

    Security Requirement: Authenticate devices

Table 1. Security requirements and threat mitigation

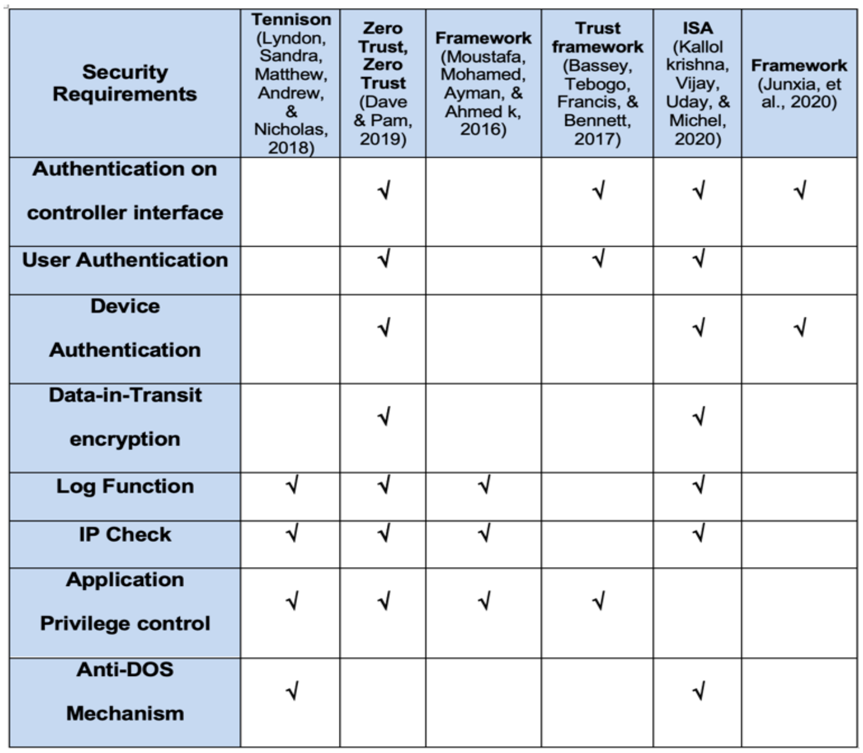


Security Mechanisms	Security Requirements	Threats Mitigation	
		Successful	Unsuccessful
TLS w/ PKI	Authentication on controller interface	√	
TLS w/ PKI	User Authentication		√
TLS w/ PKI	Device Authentication	√	
TLS v.1.2	Data-in-Transit encryption	√	
Syslog	Log Function	√	
Stateful Firewall	IP Check	√	
Stateful Firewall	Application Privilege control		√
Snort IDS/IPS	Anti-DoS Mechanism	√	

After identifying security requirements, the proposed framework is implemented. Upon implementation of the framework, the network is re-assessed to see if the security requirements were met. The table below shows the summary of the results.

4.1. Comparative Analysis of Security existing security frameworks

This section presents the comparative analysis of proposed framework with existing frameworks from previously discussed literature based on the satisfaction of SDN security requirements. The table below shows the results from different frameworks.



Security Requirements	Tennison (Lyndon, Sandra, Matthew, Andrew, & Nicholas, 2018)	Zero Trust, Zero Trust (Dave & Pam, 2019)	Framework (Moustafa, Mohamed, Ayman, & Ahmed k, 2016)	Trust framework (Bassey, Tebogo, Francis, & Bennett, 2017)	ISA (Kallol krishna, Vijay, Uday, & Michel, 2020)	Framework (Junxia, et al., 2020)
Authentication on controller interface		√		√	√	√
User Authentication		√		√	√	
Device Authentication		√			√	√
Data-in-Transit encryption		√			√	
Log Function	√	√	√		√	
IP Check	√	√	√		√	
Application Privilege control	√	√	√	√		
Anti-DOS Mechanism	√				√	

Table 2. Comparison of security requirement & proposed frameworks

Upon mapping the security requirements each proposed framework satisfies, it is seen each one has varying focus. Among the compared proposals, Cisco's Zero Trust, Zero Trust network satisfies all security requirements that are focused on this paper. However, because it utilizes Cisco SaaS solution, it is not completely vendor independent [16]. Regardless, all the compared frameworks along with the security framework proposed in this paper provides value for security assurance of Software-Defined Networks.

4.2. Latency Analysis

Network performance is a crucial factor in any environment. Different applications and services running on the network equate to higher resource utilization and, in turn, can lead to network latency [41] [42] [43]. Since security mechanisms are invasive and consume processing and storage resources, analyzing their impact on network latency and Quality of Service (QoS) is necessary. Upon inspection of ICMP ping results from different experiments done in this research, network latency from before and after implementing security mechanisms is analyzed and plotted. The graph below represents the difference in latency from H1 to H2 with a default payload of 64 bytes and no custom delay configured in the link.

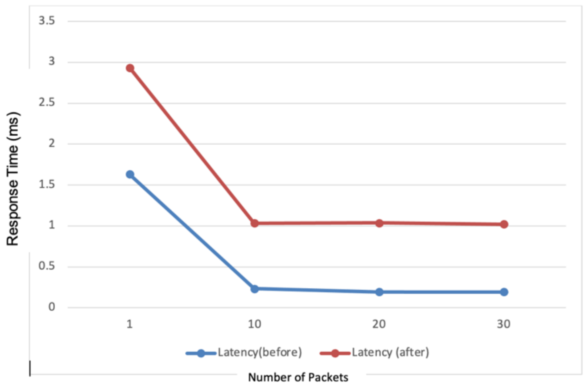


Figure 3. Latency mapping

5. Conclusion

This research proposed and implemented a security framework that focused on securing enterprise networks by implementing standard security mechanisms. The research attempted and succeeded in narrowing the gap between security requirement and availability. Since security is a constant process and has multiple vectors, the proposed framework standalone is not sufficient. However, it is a step towards the implementation of safe practices. The proposed framework can be expanded to include additional security controls. Additionally, the tests can be furthered to include "host," "network," and "application" centric vulnerability assessments and mitigation techniques. This will expand the scope of the project and set requirements for the additional security mechanisms.

REFERENCES

[1] E. E. Haleplidis, S. Denazis, J. H. Salim, D. Meyer and O. Koufopavlou, "Software-Defined Networking(SDN) : Layers and Architecture Terminology," Internet Research Task Force (IRTF), 2015.
[2] A. S. a. J. P. A. Prajapati, "Software defined network: Future of networking," in 2018 2nd International Conference on Inventive Systems and Control (ICISC), Coimbatore, 2018.
[3] MarketsandMarkets, "Software-Defined Networking Market- Global Forecast to 2025," marketsandmarkets research, 2020.
[4] Dissanayaka, Akalanka Mailewa, Susan Mengel, Lisa Gittner, and Hafiz Khan. "Vulnerability prioritization, root cause analysis, and mitigation of secure data analytic framework implemented with mongodb on singularity linux containers." In Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis, pp. 58-66. 2020.
[5] X. Lei, H. Jeff, h. Sungmin, Z. Jialong and G. Guofei, "Attacking the brain; Races in the SDN control plane," in 26th USENIX Security Symposium, Vancouver, 2017.
[6] B. Kevin, C. L. Jean and S. Chris, "Openflow vulnerability assessment," in 2nd ACM SIGCOMM workshop on hot topics in software defined networking, 2013.
[7] C. Juan, I. Jenny and V. Juan, "Security in SDN: A comprehensive survey," Journal of Netowrk and COmputer Applications, 2020.
[8] M. Nadya El, T. Ahmed and A. Maryam El, "Security analysis as software-defined security for SDN environment," in 2017 Fourth International Conference on Software Defined Systems (SDS), Valencia, 2017.
[9] T. Bakhshi, "State of the art and recent research advances in software defined networking," Wireless communications and mobile computing, vol. 2017, p. 35, 2017.
[10] D. Marc, D. Sven, K. Frank and K. hartmut, "Network Attack Detection and Defense - Security Challenges and Opportunities of Software Defined Networking," Dagstuhl Reports, vol. 6, no. 6, pp. 1-28, 2016.
[11] S. Myung-Ki, N. Ki-Hyuk and K. Hyong-Jun, "Software-defined networking (SDN): A reference architecture and open APIs," in 2012 International Conference on ICT Convergence (ICTC), Jeju, 2012.
[12] J. Yosr, M. Taous and D. Mourad, "A survey and a layered taxonomy of Software-Defined Networking," IEEE communciation surveys & tutorials, pp. 1955-1980, 2014.
[13] L. Wenjual, M. Weizhi and K. Lam, "A survey on openflow-based Software defined networks/; securit challenges and countermeasures," journal of network and computer applications, vol. 68, pp. 126-139, 2016.
[14] L. Junxia, C. Jinjin, K. Fazlullah, R. Ateeq Ur, B. Venki, S. Jeangfeng and P. Venu, "A secured framework for SDN-Based Edge Computing in IoT-Enabled Healthcare System," IEEE Access, vol. 8, pp. 135479-135490, 2020.
[15] F. Lyndon, S.-h. Sandra, B. Matthew, W. Andrew and R. Nicholas, "Tennison: A distributed SDN framework for scalable network security," IEEE Journal on selected areas in communications, vol. 36, no. 12, pp. 2805-2818, 2018.
[16] D. Dave and R. Pam, "Zero Trust, Zero Touch," Cisco systems, San Jose, 2019.
[17] A. Moustafa, R. Mohamed, A.-H. Ayman and A.-S. Ahmed k, "A framework for security enhancement in SDN-Based Datacenters," in 2016 8h IFIP International Conference on New Technologies, Mobility and Security (NTMS), Larnaca, 2016.
[18] I. Bassey, K. Tebogo, L. Francis and K. Bennett, "Trust establishment framework between SDN controller and applications," in 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Kanazawa, 2017.
[19] K. Kallolkrishna, V. Vijay, T. Uday and h. Michel, "Towards a Dynamic Policy Enhanced Integrated Security Architecture for SDN Infrastructure," in NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, Budapest, 2020.
[20] H. Ali, E. imad H, C. Ali and K. Ayman, "SDN Security Plane: An Architecture for Resilient Security Services," in 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW), Berlin, 2016.
[21] S. Yue, D. Fangfanf and Y. Zhiguo, "An enhanced security framework of software defined network based on attribute-based encryption," in 4th International Conference on Systems and Informatics (ICSAI), Hangzhou, 2017.
[22] P. Bhushan, P. Sonal, S. Zia and S. Anupam, "Framework for securing SDN southbound communication," in International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, 2018.
[23] S. Ahmed, R. Ahmed and S. Abdallah, "On the security of SDN: A completed secure and scalable framework using the software-defined perimeter," IEEE access, vol. 7, pp. 146577 - 146587, 2019.
[24] Thapa, Suman, and Akalanka Mailewa. "The Role of Intrusion Detection/Prevention Systems in Modern Computer Networks: A Review." In Conference: Midwest Instruction and Computing Symposium (MICS), vol. 53, pp. 1-14. 2020.
[25] C. Juan, I. Jenny and V. Juan, "Security in SDN: A comprehensive survey," Journal of Network and Computer Applications, vol. 159, no. 102595, 2020.

[26] S. S. a. G. Gu, "Attacking Software-Defined Networks: A First Feasibility Study," in second ACM SIGCOMM workshop on Hot topics in software defined networking, 2013.

[27] K. Suleman, A. B. Mustapha, W. A. W. Ainuddin, G. Abdullah and A. Ahmed, "Understanding Link Fabrication Attack in Software Defined Network using Formal Methods," in 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, 2020.

[28] H. Xinli, S. Peng, y. liu and X. fei, "Towards trusted and efficient SDN topology discovery: A lightweight topology verification scheme," Computer Networks, vol. 170, 2020.

[29] S. S. Hayward, S. Natarajan and S. Sezer, "A Survey of Security in Software Defined Networks," IEEE Communications surveys and tutorials, pp. 623-654, 2016.

[30] D. B. Rawat and S. R. Reddy, "Software Defined Networking Architecture, Security and Energy Efficiency: A Survey," IEEE communications surveys and tutorials, pp. 325-346, 2017.

[31] Mailewa Dissanayaka, Akalanka, Roshan Ramprasad Shetty, Samip Kothari, Susan Mengel, Lisa Gittner, and Ravi Vadapalli. "A review of MongoDB and singularity container security in regards tohipaa regulations." In Companion Proceedings of the10th International Conference on Utility and Cloud Computing, pp. 91-97. 2017.

[32] Ryu development team, "RYU Documentation," Nippon Telegraph and Telephone Corporation, 2021.

[33] Open Networking Foundation, "Openflow Switch Specification Version 1.5.1," Open Networking Foundation, 2015.

[34] K. Pakapol and S. Yuthapong, "Implementation of SDN Stateful Firewall on Data Plane using Open vSwitch," in 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE), Nakhonpathom, 2018.

[35] Dissanayaka, Akalanka Mailewa, Susan Mengel, Lisa Gittner, and Hafiz Khan. "Dynamic & portable vulnerability assessment testbed with Linux containers to ensure the security of MongoDB in Singularity LXCs." In Companion Conference of the Supercomputing-2018 (SC18). 2018.

[36] Dissanayaka, Akalanka Mailewa, Susan Mengel, Lisa Gittner, and Hafiz Khan. "Security assurance of MongoDB in singularity LXCs: an elastic and convenient testbed using Linux containers to explore vulnerabilities." Cluster Computing 23, no. 3 (2020): 1955-1971.

[37] U. Raja Majid Ali, P. Zeeshan and D. Keshav, "Snort Based Collaborative Intrusion Detection System Using Blockchain in SDN," in 2019 13th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), Island of Ulkulhas, 2019.

[38] Mohammad, Rajiullah; Reine, Lundin; Anna, Brunstrom; Stefan, Lindskog, "Syslog performance: Data modeling and transport," in 2011 Third International Workshop on Security and Communication Networks (IWSCN), Gjovik, 2014.

[39] Open Networking Foundation, "Security Foundation Requirements for SDN Controllers," Open Networking Foundation, Palo Alto, 2016.

[40] S. Dominik, "Secure communication between openflow switches and controllers," in AFIN 2015 : The Seventh International Conference on Advances in Future Internet, 2015.

[41] A. Bachar and G. Gheorghita, "A Performance Evaluation of Security Mechanisms for Web Services," in 2009 Fifth International Conference on Information Assurance and Security, Xi'an, 2009.

[42] Simkhada, Emerald, Elisha Shrestha, Sujan Pandit, Upasana Sherchand, and Akalanka Mailewa Dissanayaka. "Security threats/attacks via botnets and botnet detection & prevention techniques in computer networks: a review." In The Midwest Instruction and Computing Symposium.(MICS), North Dakota State University, Fargo, ND. 2019.

[43] Shetty, Roshan Ramprasad, Akalanka Mailewa Dissanayaka, Susan Mengel, Lisa Gittner, Ravi Vadapalli, and Hafiz Khan. "Secure NoSQL based medical data processing and retrieval: the exposome project." In Companion Proceedings of the10th International Conference on Utility and Cloud Computing, pp. 99-105. 2017.