EXPLORING THE SECURITY OF SOFTWARE DEFINED NETWORK (SDN)

Authored by

Prabhjot Kaur (144113)

Shiv Patel (143792)

Sanjana Mittal (144509)

Surbhi Sharma (144469)

Research Project

Submitted to Faculty of Graduate Studies,

Concordia University of Edmonton

In Partial Fulfillment of the Requirements for the Final

Research Project ISSM681(R)

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Alberta

Advisor: Dr. Sergey Butakov (sergey.butakov@concordia.ab.ca)

June 2021

EXPLORING THE SECURITY OF SOFTWARE DEFINED NETWORK (SDN)

Prabhjot Kaur

Shiv Patel

Sanjana Mittal

Surbhi Sharma

Approved:

Sergey Butakov

Sergey Butakov
Primary Supervisor

Date: June 23, 2021

Patrick Kamau

Patrick Kamau, PhD, MCIC, PChem.
Dean, Faculty of Graduate Studies

Date: June 23, 2021

Table of Contents

List of Figures

Exploring the security of Software Defined Network (SDN)

Prabhjot Kaur
Student ID #:144113
pkaur16@student.concordia.ab.ca

Shiv Patel
Student ID #: 143792
spatel3@student.Concordia.ab.ca

Sanjana Mittal
Student ID #: 144509
smittal@student.concordia.ab.ca

Surbhi Sharma
Student ID #: 144469
ssharm18@student.concordia.ab.ca

Advisor: Dr. Sergey Butakov
sergey.butakov@concordia.ab.ca

Department of Information Systems
Security Management
Concordia University of Edmonton,
Edmonton T5B 4E4, Alberta, Canada

***Abstract*—** Software Defined Networks have a centralized nature due to which the attackers may try to compromise them to jeopardize the whole network security. The SDN controller is the center point for connections between the applications and the network, becomes the potential candidate for network attacks such as man-in-the-middle, distributed denial of service (DDoS) attacks. In this paper, the SDN infrastructure is exposed to various DDoS attacks and then the results are noted based on the severity of the attacks. In a nutshell, this paper studies the potential security vulnerabilities of unencrypted communication in the northbound and southbound channels. The experiment's conclusion established that a DDoS attack on one VLAN affected the services of another VLAN. The VLANs were built to segregate traffic without inter-VLAN contact, but the massive amount of traffic produced by a DDoS attack on one VLAN strained the controller's resources, delaying the response of legal traffic from other VLANs and resulting in a Denial of Service attack against that VLAN.

Keywords—Software Defined Network (SDN), Controller, Northbound Interface (NBI), Southbound Interface (SBI), Application Programming Interface (API), OpenFlow, Security

I. INTRODUCTION

Software-Defined Network (SDN) has started emerging in the IT Industry. In traditional Networking, the hardware and software are used to transfer the data across the switches and routers, while Software Defined Networking segregates the control plane-where the network is managed and the data plane-where the traffic is directed through routers and switches [1]. The controller has software installed to handle and manage the network traffic that will route through a series of switches and routers of the data center. Virtualizing the SDN network helps to dynamically divide the traditional overall network, dedicate a segment of the overall network to a specific application, and apply specific security policies to each network. SDN is more agile and manageable due to the software-based controller and can adapt to multiple use cases. The significant features of SDN are:

a) Microsegmentation to enhance Security: Microsegmentation allows dividing the network and isolating each of them securely such that if one of the networks is under attack, others are safe. It provides security at a granular level and more control over the network.

b) Centralized Control: SDN offers centralized control to the data plane and application plane. It makes easy network management of the physical and virtual resources from one centrally controlled location. Network Administrators can centrally manage the resources as per the security policies and information[10].

c) Virtualization offering Agility: Virtualization allows the developers to control the allocation of resources at different locations, centrally from the SDN Controller.

d) Easy Programming of Networking Devices: In SDN, the northbound interface allows the connections between the controller and the various applications. The programmable interface helps the developers directly program the network devices, unlike the traditional network where the devices are programmed with vendor-specific configurations and protocols [2].

e) Less Deployment and Operational cost: In SDN, switches, routers, and other networking devices are centrally controlled and managed, reducing the overall setup, maintenance, and operational cost.

f) SDN Cloud Abstraction: SDN can manage networking components such as large data center platforms [2]. SDN allows a greater level of automation in the cloud, improving configuration, provisioning, and management.

A. SDN Architecture

Software-defined Networking segregates the data plane and the centralized control plane for running multiple types of applications. In a standard architecture, SDN is divided among three different planes: Application Plane, Control Plane, and Data Plane, as shown in figure 1 below. All the layers are separated and isolated in this figure, but they interact using the northbound and southbound interface. In the following paragraphs of this section, SDN components have been briefly discussed:

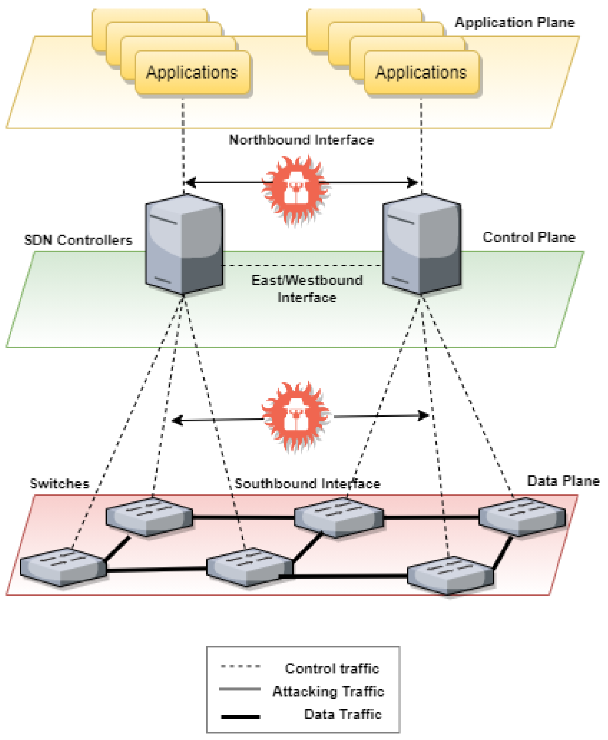


Figure 1: SDN Architecture and potential location of the attackers on the interfaces

a) Data Plane: The data plane of the Software-Defined Network is also referred to as the Forwarding Plane. It includes network devices such as routers, bridges, switches, etc., which are programmable and managed by the SDN Controllers [3]. Instead of working as a vendor-specific routing device, the software-based devices process and forward the data traffic as per the OpenFlow controller's instructions.

b) Control Plane: The control plane is the intellect or the processor of the Software-Defined Networking. Control Plane controls both the data plane and application plane. It manages the flow control in the networking devices through Southbound Interface while managing its interaction with the Controller through the Northbound Interface [3]. All the functionality of the control plane is software-based, which allows dynamic configuration and easy management. For example, a network administrator can update flow table entries of data packets through the centralized control without making any changes at the individual switches. The administrator can also prioritize or block certain data packets.

Almost all the SDN Control Plane offers various networking services such as statistics management, routing module, device management [3], firewall management, etc.

c) Application Plane: Application Plane is the external interface that allows the communication between the Application Layer and the other SDN Plane. The application interacts with the networking devices bypassing their requirements through the Northbound Interface [4]. Applications are also used to control, manage, manipulate, and set the policies' underlying physical and virtual network devices. Additionally, it consists of applications utilizing the network services such as Network Security, Access Control Management, Load Balancing, Quality of Service, Traffic Engineering, intrusion detection systems, virtualization services [4], etc.

d) Northbound Interface: Northbound APIs are the upper part of the SDN and communicate between the controller and the application layer components [5]. It allows the network provider to utilize the interface to build the SDN or regain information using GUI or API. The northbound API may be employed in multiple ways, like adding a brand new VLAN on your switches, checking the topology, automatically configuring the IP address, providing routing, adding or deleting a virtual machine, etc. [6]. Northbound API also allows an easy interaction of the SDN Controller with the firewalls, load balancers, software-defined security services, and cloud devices.

e) Southbound Interface: Southbound APIs are an OpenFlow (or others such as Cisco OpFlex, CLI) protocol specification used to communicate between the controllers and data plane devices [5]. Open-Flow is the standard Southbound interface that creates a secure channel between the Open Flow Controller and Open Flow Switch [7]. It is a Southbound API that can be more responsive to real-time traffic demands and allows network administrators to remove or add the network devices' routing table entry [5].

The Southbound interface's main challenge arises from vendor-specific network devices [3], but it is managed because of the open and standardized southbound API interface.

f) East/Westbound Interface: East/West-bound Interface is used to communicate among the distributed SDN Controllers. It also monitors to ensure that the controllers are up and working.

The layout of a software-defined network is conducive to innovation. Apart from SDN Controllers' various benefits such as centralized control, easy network device management, traffic engineering, and configuration, it is vulnerable to security attacks. This paper has explored various ways in which an SDN controller can be exploited using DDoS(Distributed Denial of Service) attacks.

II. Literature Review

A. *Security Issues related to SDN*

Being the network's processing unit, the SDN controller enables the connection between the applications and network devices and decides the flow and control of packets across the data plane. Therefore, it becomes a potential candidate for a security attack and can badly affect the network. There are many vulnerabilities in the SDN controllers, such as weak encryption, information disclosure, weak authentication, etc. [12] , which leads to various attacks, including DoS, Spoofing, Tampering, Elevation of Privileges, DDoS, etc.

According to the literature review of various articles, the security issues from which the SDN must be secured are as follows:

a) Being a network service backbone, the control plane acts as a vulnerable point in the SDN [13]. An attacker can forge the victim's IP address through network monitoring to get trusted by a switch and embed malicious code through loopholes to exploit the system [14] to launch the DoS attack.

b) XML External Entity issue(XXE): Open Daylight stores some configuration files in the controller related to the southbound interface's network devices. It is vulnerable to an XML External Entity attack when the NETCONF protocol is used [15]. This vulnerability can produce information disclosure of those configuration files of the controller [15].

c) SQL Injection Attack: In this attack, an attacker can SQL inject the Open Daylight component database(SQL Lite) without authenticating itself to the ODL controller or interface application [16]. This attack can result in information disclosure of sensitive data such as passwords, SIN, statements, etc.

d) Forwarding Device Attack: In this attack, the malicious entity generates excess traffic from the data plane devices such as switches to overwhelm the controller [17]. This can affect the communication between the southbound, northbound interface, and the processes. Also, this attack can overload the network resources with the spoofed data packets and launch the DoS attack.

e) Information Disclosure and Tampering: SDN Controllers have the possibility of information disclosure due to the unencrypted channel between the controller and applications in the northbound interface [15]. It only uses HTTP instead of HTTPS for the interactions. Moreover, the southbound interface communication is also not encrypted using TLS [15]. This makes it vulnerable to information disclosure and tampering.

f) Man in the middle attack (MitM): Most controllers are vulnerable to the tampering of the data due to the unsecured flow of data packets between the controller and the northbound applications [18]. The attacker can perform a simple ARP Spoofing, perform a MITM attack, and alter the packets' content to destroy the unencrypted communication channel [15].

g) Spoofing: In SDN Controllers, spoofing is likely to happen because of the absence of an authentication mechanism in both the northbound and southbound interface [15]. An attacker with a spoofed MAC address similar to the real machine can alter the configuration and attack the network [15]. Additionally, a controller can accept a packet from a switch without performing authentication on it.

h) Open Programmable API: On the controller's NBI and SBI, logging is default disabled while communication between the APIs, controller, and switches. This weak programmability feature of the SDN Controller has possible repudiation chances to occur [19].

B. *DoS/DDoS attacks on the controllers and their protection*

DOS/DDOS attacks are the most challenging threats to any organization's network. Attackers attempt this type of attack in multiple ways to make the network services unavailable by choking links, overwhelming servers, and flooding the buffer of network devices with illegitimate traffic.

In this section, DoS/DDoS attacks on the SDN controllers and their protection measures have been discussed. According to the literature review of various research papers, the following are the types of DDOS attacks on the controller:

a) *Flooding Packet-in message:* Packet-in messages are used by the virtual switch to get the new packet controller's flow rule. The attacker floods the multiple packets to switch with a spoofed IP address, which forces the switch to send the flow rule request in bulk and makes the controller busy to entertain the fake flow requests[11]. This denial of service attack is carried on the virtual switch, but it affects the controller due to centralized control.

b) *Saturating Controller:* The Controller creates a queue to cater to the multiple flow request, but an attacker generates numerous fake packets, which results in degrading the controller performance by utilizing the controller resources [12].

c) *Southbound API's Congestion:* Virtual switch always sends some part of the packet along with packet-in messages to a controller for the new rule. Once the switch buffer gets full, it sends the entire packet with a packet-in message to the controller via southbound API. An attacker could generate the fake flows to switch, and due to the full buffer switch forwards the huge packets over single links, this could create congestion by utilizing the bandwidth and makes it unavailable [13]. This attack makes the southbound interface unavailable, which breaks the connection between the controller and data plane devices.

In this section, the protection mechanism on the northbound and southbound interface has been discussed, capable of mitigating the potential causes that can create the DoS/DDoS attack scenario. SDN Controllers can be protected from the DoS/DDoS attacks in the following ways:

a) Protection mechanism on the northbound Interface: The SDN controller combines with the application plane to form a Northbound Interface to enable interaction of applications with the controller and data plane devices. However, Northbound APIs are vulnerable to malicious intrusion due to the connectivity to the application plane. The architecture of Northbound APIs could be created using a variety of different technologies and programming languages. The vulnerability of such programming languages will be carried forward and acts as a potential for malicious activity on the controller. In other cases, an attacker might create their policies by exploiting a vulnerability of northbound API and gain control of the SDN environment.

Some of the protection mechanisms against DoS/DDoS attacks that can be implemented on the Northbound interface of the SDN are as follows:

- Entropy: An SDN controller can control the entropy and bandwidth of each packet passing through it [20]. The author suggests using entropy to evaluate traffic and enforce mitigation strategies. It will help the controller to filter out the malicious user and restrict them.
- OAuth: It is used as an authentication framework in the SDN controller northbound interface utilizing the tokens

and the authorization [21]. An authentication server is a mechanism where the API key and secret are exchanged for an access token, and the user is not involved in the authentication process. The access token is an identifier dependent on the network policy.

- The third-party installation: Tools such as iftop are used to evaluate the incoming data packets' bandwidth with the conditions of a DDoS attack [22]. The device shows how long an attacker can launch a DDoS attack [22]. Hence, using these kinds of third-party tools restricts network access to the network, preventing an intruder from gaining access to the server.
- Self-Signed Certificates: In this case, the controller requires a legitimate server certificate called the database certificate [23]. The controller is signing a certificate, and the certificate authority is signing it (CA), which ultimately enhance the integrity and prevent the DDoS attack.
- The northbound interface can prevent the DoS attack using Rate Limiting, Event Filtering, Packet Dropping, Rule Timeout adjustment, etc. [19]. It can also be managed by implementing an authentication mechanism at the application interface.
- Defense4All: In ODL, the Defense4All mechanism can remove the threat of denial of service in the controller [15]. It secures the northbound, southbound processes and data from the network attacks.

b) Protection mechanism on the Southbound Interface: Southbound interfaces ensure how the data plane should exchange information with the SDN controller to adjust the network. The OpenFlow needs the channel between controllers and switches to be secured using TLS. This invites vulnerabilities as it opens the security holes. In SDN, the Southbound Application interface is necessary to get the control plane's instruction to forward the data plane devices' packets. However, the attacker could exploit Southbound APIs' vulnerabilities or data plane devices to attack and make it unavailable. Also, the switch buffer could be flooded by fake traffic generated by an attacker to saturate the buffer memory and flood the entire packet to the controller from the southbound interface [13]. This attack raises a communication issue in the southbound interface.

The security keys against DoS attacks in the SDN architecture from the southbound interface is as follows:

- AVANT-GUARD: It is an SDN key solution against DoS attacks in the framework where the attacker uses a spoofed IP address[18]. It defends against the saturation of controller and communication overhead in the Southbound interface. It solves the issues by limiting the interaction between the data plane and control plane with the connection migration module's help. Another module, Actuating trigger, is implemented on data plane devices to collect packet and network information.
- Another author in the article [19] also addresses the communication overhead in a southbound interface by implementing a 3-phase solution called state sec. This solution is implemented on the switch is used to detect and mitigate the DOS/DDOS attacks. Those three steps are as follows:
  a) Monitoring: In this step, the switch uses stateful programming to monitor the traffic based on the port number and IP address of both source and destination.
  b) Detection: In this step, traffic is being analyzed to differentiate between fake and legitimate traffic by detecting anomalies with an entropy-based algorithm.
  c) Mitigation: Rate-limiting is being used to mitigate the attack after detecting the anomaly in the traffic.

The SDN is always the key target for the attackers because it is the primary point for decisions in a network and a primary point of failure. Hence, security is the main aspect to be considered. The above sections clearly state the vulnerabilities that are bringing down the unlimited benefits of SDN. SDN is beneficial in removing multiple layers of a firewall with just one layer but, on the other hand, also exposes layers of susceptible network skin ripe to attack. To protect the attacks, reducing the exposures by hardening the controllers and protocols will be a short-lived solution, but understanding the vulnerabilities and applying a security layer will reduce most of the attacks.

Additionally, it is essential for the SDN controller's security to fend off malicious attacks and unintentional changes. Therefore, this practical research will contribute to the existing knowledge base around the technology and improve SDN controller security aspects. This improvement would encourage more extensive use of the technology in cloud computing, wide area networks, mobile and wireless technologies. Specifically, the research's security recommendations will help the organizations securely manage the controllers and quickly respond to evolving business requirements. Thus, this research will provide a clear view of helping Canadian IT, mobile Networking, and small businesses achieve efficiency, scalability, agility, less operating and implementation cost, and enhanced configurations for network management.

III. Experimental Setup

A. Methodology

This research emphasizes the experimental and studies analysis of the vulnerabilities of SDN controllers. While conducting the analysis, the existing SDN Controllers' backdoors have been discovered and exploited to implement the Denial-of-Service attack successfully. The exploitation of these backdoors helped to measure the impact of the attack on an SDN controller with the VPN, VLAN and an encrypted communication channel. Moreover, the performance of the SDN Controller under the DoS attack has been calculated and analyzed. The following methodology has been followed:

Step 1: After analyzing various available resources related to SDN vulnerabilities, a testbed is created for performing the experimental research. It consists of multiple virtual machines based on the Ubuntu OS platform having an ODL controller (https://docs.opendaylight.org/en/latest/downloads.html), mininet (http://mininet.org/download/), and two attackers in action. Rapid Access cloud (https://rac-portal.cybera.ca/users/sign_in) is used to host all the machines in an isolated manner.

Step 2: In this research, a self-signed certificate is implemented on the ODL server to ensure that the northbound communication channel is encrypted. Additionally, the VLAN and VPN are also implemented to add an advanced security layer to the infrastructure.

Step 3: ODL controller is bombarded with an excess of requests from the attacker's virtual machines. Cbench and Apache benchmarking tools have been used to measure the effectiveness and throughput of the DoS attack.

B. *Experiment*

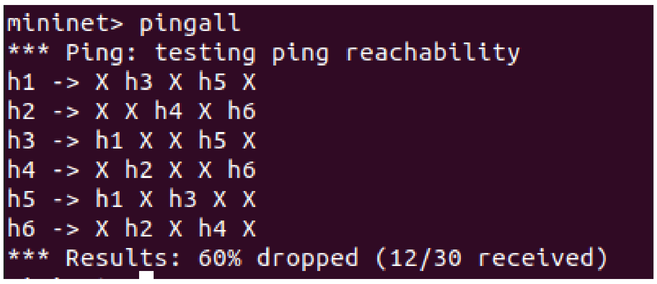
In this research, DDoS attacks have been performed on the OpenDayLight(ODL) controller using standard testing tools such as hping3, LOIC & Scapy Script Attack. Moreover, the Cbench has been used to generate traffic across the victims and the attackers. Usually, numerous hosts and massive topology is required to launch the DoS attack on the victim. However, in contrast to the real-world attacks, this research project deliberately involved a less complicated topology in studying and analyzing the DoS attack's impact on SDN. As a controller, OpenFlow-based ODL has been used due to its programmability and adaptive features.

Step 1: Implementing security at ODL

In SDN architecture, the controller is the central unit that manages the entire operations in a software-defined network. The controller consists of several northbound and Southbound API to manage the network, so implementing security to the controller is at most priority.

- The one way to secure the controller is by securing access to it. In our test environment, HTTPS has been implemented in the ODL controller using Java Keystore to ensure secure access to the API's and controller. Java Keystore is a container of Security certificates, mainly consist of authorization or public-key certificates. The Java-based application uses Java Keystore for encryption and authentication over HTTPS. Due to Java-based environment, Keystroke has been used as the solution to protect the controller access along with the self-signed certificate and HTTPS 8443. Once the certificate is created, HTTPS is enabled and a path to Keystore is provided. After that, the ODL controller is run in the web browser with (https://(controller IP):8443/index.html#/login). After running the above command, click on accept and add the certificate

  After accepting the risk, a login window will prompt and log in to the device securely with the given password and username.
- Network segmentation in the ODL network also enhances security by limiting the attacks like DDOS to one network without affecting the other. One of the ways to achieve network segmentation is through VLAN (virtual local area network). VLAN allows a network admin to put a host in multiple broadcast domains which restricts the host from different broadcast domains to communicate with each other. In this ODL controller, python code is created using Southbound API like Netconf, mininet to create VLAN on switches and control them by the ODL controller. After the installation, 6 hosts were configured in the two different VLANs, and connectivity was tested. The test results showed that the host h1, h3 and h5 of VLAN 200 are not able to reach the host h2, h4 and h6 of VLAN 300. Hosts of VLAN 200 and 300 were not able to pass traffic between each other (ping) as Figure 2 shows.



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X h5 X
h2 -> X X h4 X h6
h3 -> h1 X X h5 X
h4 -> X h2 X X h6
h5 -> h1 X h3 X X
h6 -> X h2 X h4 X
*** Results: 60% dropped (12/30 received)
```

Figure 2: Ping response from the host in different VLAN.

Later, an attack was initiated in the first VLAN network to test whether it has any effect on another VLAN network.

Step 2: Attacking procedures

a) Hping3: Hping3 is a packet generator and TCP/IP analyzer used to simulate the DoS attack on the SDN controller *[8]*. This penetration tool has been used to create TCP SYN flood on the ODL web server. In this DDoS attack, the following hping3 attributes are used: -c (packet count), -S(SYN packets), -p(Port Number),-w(winsize(default 64)), -i(wait interval).

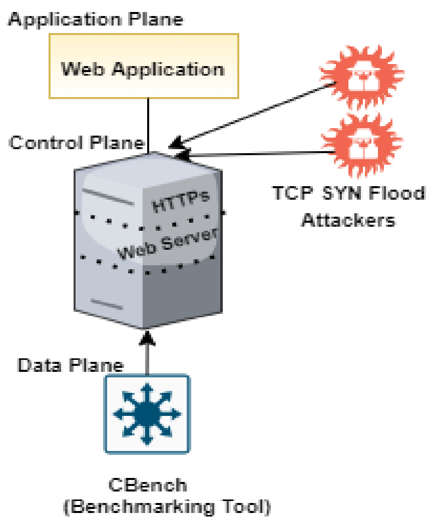


Figure 3: Network Topology for SYN flood attack using hping3

Attack Description: In this attack, CBench has created some fake switches that can send fake IP packets to the target controller IP address. It is a benchmarking tool designed to estimate the performance of OpenFlow SDN controllers. Simultaneously, the hping3 command has been used from the attacker machine to bombard the TCP packet's target controller. This tool helps to simulate a DDoS attack on the ODL Controller by affecting the bandwidth and increasing the response time.

b) LOIC Attack: Another penetration tool used for network stress testing and denial of service and distributed denial of service (DDoS) attacks. DDoS attacks use this tool to overwhelm an attacker's target's network with junk TCP, UDP, and HTTP request GETs. *[9]*

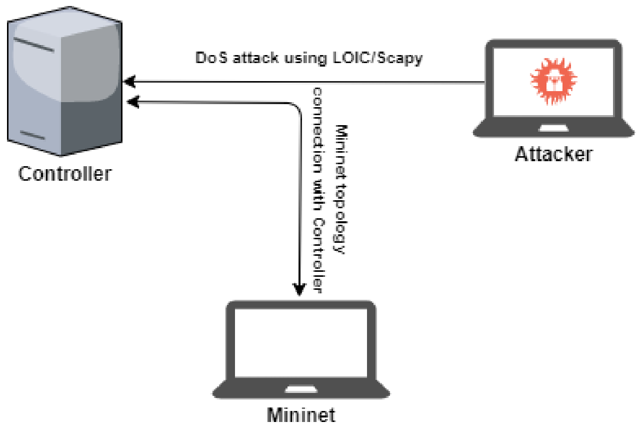


Figure 4: Network Topology for LOIC attack

Attack Description: Attack simulation can be launched by first entering the IP or URL into LOIC, indicating a TCP, UDP, or HTTP flood. The TCP and UDP flood modes may send packets to various ports, while the HTTP flood mode sends a continuous stream of GET requests. LOIC builds connections to the targeted server and then bombards the server with requests until the server becomes overwhelmed and cannot respond to legitimate requests. It must be remembered that LOIC users cannot route traffic through proxies. Due to their IP addresses being readily available, they are easily traceable [9].

c) Scapy script Attack: Scapy is software used for sniffing, sending, forging, and spoofing. Scapy is a powerful tool used to decipher several protocols, manipulate packets on the network, and send them to the network and receive responses. Scapy has many advantages over other network analysis methods. Using Scapy to construct a raw packet will take less time than writing equivalent code in C. The Scapy tool acts on matching packets and unmatched packets. It can also be used to execute ARP poisoning and many other attacks [10].

Scapy has been used to construct a TCP packet with the destination port equal to 6653, and all the other parameters were left unchanged. The / operator is used to connect different protocol sublayers. On top of IP, TCP is also stacked on top of an Ethernet. Subsequently, the generated packet is printed in the shell. Other important functions are:

d) Fragmentation script Attack using scapy: Many computers fall for this attack because they are configured to accept packets of 65,535 bytes, which exceeds the maximum IPv4 limit, though the attacker-defined packet size.

Next, Start the packet sniff for the victim virtual machine and maintain a continuous transmission of 65,565-byte fragments. Now, conduct an overview of the packet summaries present on the victim virtual machine. If the packet is sent in its entirety, then no overflow would occur, but the device could crash if the target machine reassembled it.

Step 3: Results

As a result of the scapy attack, the packets captured at the victim's computer are not able to reassemble. Consequently, it results in data packets colliding rapidly and overloading the victim's servers, causing them to fail.

Similarly, in the LOIC/scapy attack, the victim server is overwhelmed and made unavailable. There are two types of LOIC attacks: the TCP and UDP designs spread messages and packets along specified channels, while the HTTP flood type floods multiple HTTP requests into the target's system. With a scapy attack, the victim's computer is bombarded with an infinity of GET requests.

After generating the floods, Apache benchmarking tool was used to measure the performance of ODL with a workload of 16-20 switches. To calculate the performance, measurements were taken before and after every LOIC attack (TCP, UDP, HTTP) as shown in the graphs.

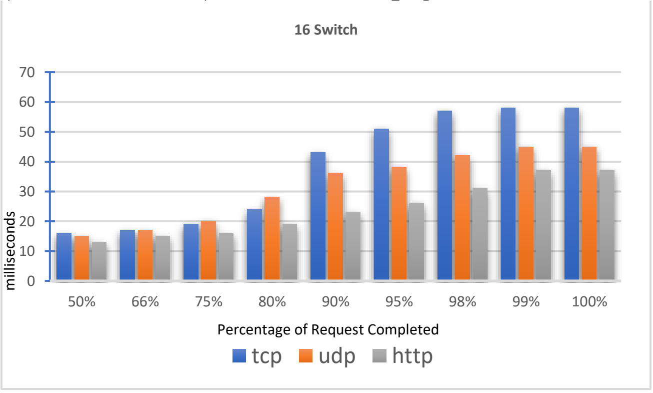


Figure 5: workload of 16 switches without DDoS attack

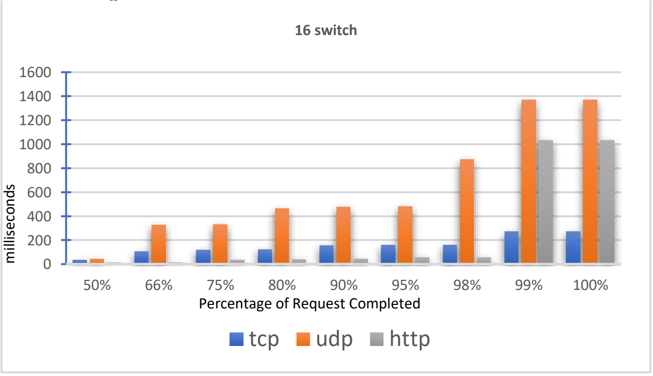


Figure 6: workload of 16 switches with DDoS attack

As security is critical to the operation of any electronic system or network. ODL is no exception; similarly, there will be a security breach if there is no security in our experimental setup. Therefore, firstly an SSL certificate was implemented on the ODL and the benchmark was measured as soon as possible without the DDoS attack, as shown in Figures 5 & 7, where TCP and UDP exhibit greater more milliseconds to complete the request than HTTP, as fig 5 indicates that it took approximately 18ms to complete 50% of requests. It increased by approximately 60ms to complete 100 percent of the request. The next step was to attack the ODL with TCP, UDP, and HTTP requests using the LOIC attack tool, as illustrated in fig 6. Following the attack, UDP and HTTP required more milliseconds to complete 100% of the request than TCP.

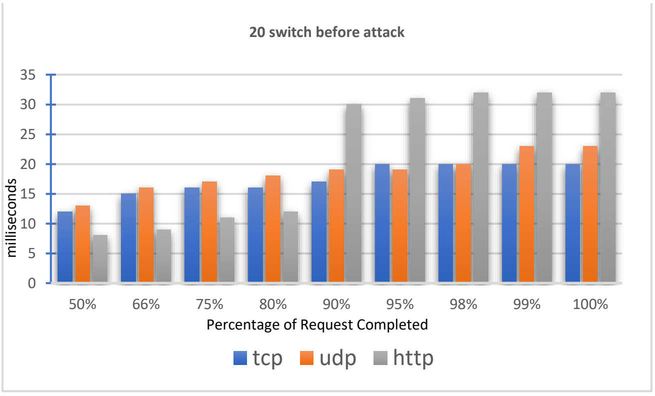


Figure 7: workload of 20 switches without DDoS attack

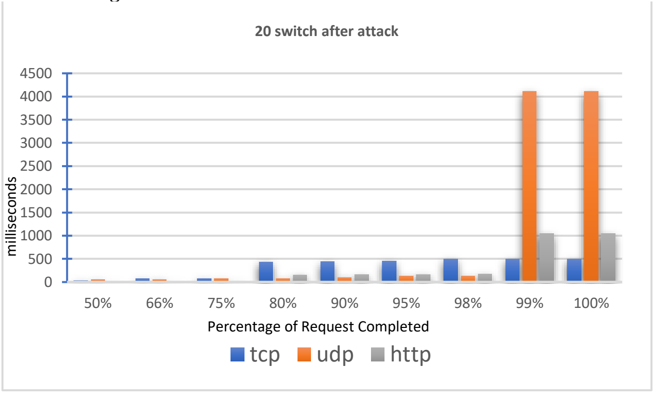


Figure 8: workload of 20 switches with DDoS attack

As an outcome of the results, it was determined that DDoS attacks are still thriving after implementing SSL, but at least there is a confidentiality provision.

Numerous DDoS defense mechanisms based on SDN include mitigation strategies. Dumping packets, restricting ports, and rerouting traffic are all frequently used control measures in SDN. There are more controls like changing the IP and MAC Address implementing VPN and VLAN for secure communication. While for Faster mitigation dumping packets or restricting the port is best because it simple and can completely stop the attack source. To get a sense of the DDoS defense mechanisms previously mentioned, all of them are dedicated to computing a base level threshold that acts as a baseline for attack detection.

In modern-day data centers, they are various applications with different requirements for networks and services. Specific critical applications, for example, have stringent uptime and availability requirements. As in the case of these applications, rapid detection of an attack is critical. Some applications have uptime requirements that are pretty low in comparison. The applications in this class can tolerate some network latency and tolerate a high rate of false positives, resulting in service denial to legitimate users. The existence of diverse applications requiring customizable solutions that respond to attack threats all come together to make it a requirement for a large-scale data center to include many applications and an array of varying levels of security sensitivity.

e) DDoS attack in Segmented network: Vlans are implemented to limit the access to a certain group which improves the security, performance and flexibility. The same concept is implemented in an SDN network to test the performance and security of the controller by attacking the host on one VLAN and monitor the impact on another.

The test environment is created in mininet and the topology used is shown in figure 10.

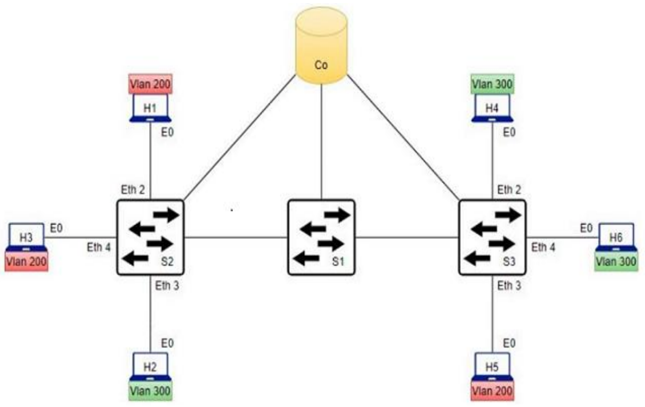
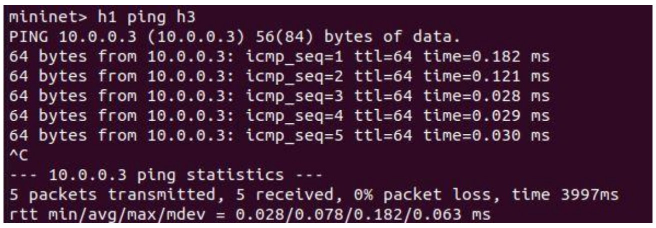


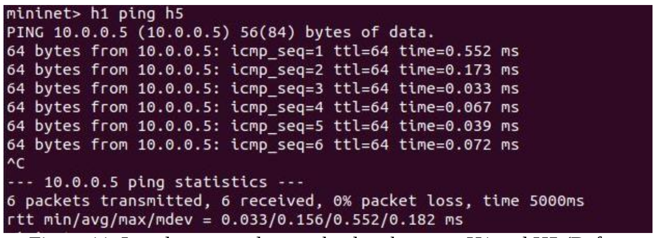
Figure 9: VLAN attack topology

As shown in figure 9., multiple hosts are placed in VLAN 200 and 300. The ping response and packet drops between the hosts of VLAN 200 have also been tested before initiating the attack in VLAN 300. In the results, good ping response and no packet drops between the hosts of VLAN 200 can be seen. The same can be seen in figure 10 and figure 11.



```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.182 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.121 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.028 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.029 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.030 ms
^C
--- 10.0.0.3 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3997ms
rtt min/avg/max/mdev = 0.028/0.078/0.182/0.063 ms
```

Figure 10: Low latency and no packet loss between H1 and H3 (Before attack)

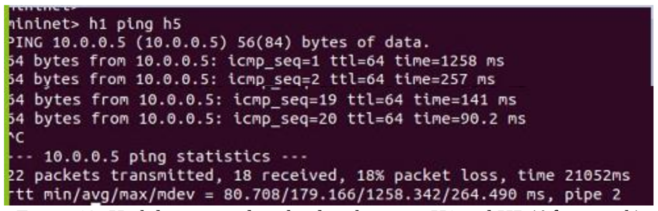


```
mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=0.552 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=0.173 ms
64 bytes from 10.0.0.5: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 10.0.0.5: icmp_seq=4 ttl=64 time=0.067 ms
64 bytes from 10.0.0.5: icmp_seq=5 ttl=64 time=0.039 ms
64 bytes from 10.0.0.5: icmp_seq=6 ttl=64 time=0.072 ms
^C
--- 10.0.0.5 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5000ms
rtt min/avg/max/mdev = 0.033/0.156/0.552/0.182 ms
```

Figure 11: Low latency and no packet loss between H1 and H5 (Before attack)

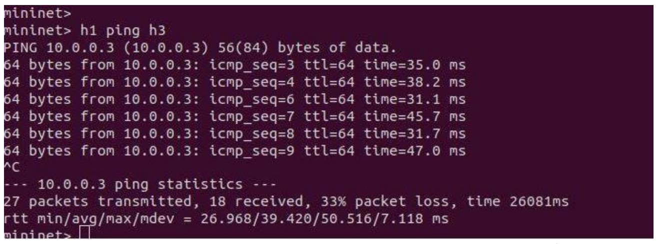
After successful ping results and latency tests between the hosts in VLAN 200, DDos attack was initiated in VLAN 300 by making H2 as a victim and H4 and H6 as the attackers. Hping3 was used in hosts H4 and H6 to generate the traffic with random source IP address, which started sending large packets towards the H2. Due to no existing flows in the switch, the packets were forwarded to the controller by the switch to get the flows for forwarding the packets destined towards H2.

Attack was underway and the reachability of hosts in VLAN 200 was checked and packet loss with high latency was found. The same can be seen in figure 12 and figure 13.



```
mininet> h1 ping h5
PING 10.0.0.5 (10.0.0.5) 56(84) bytes of data.
64 bytes from 10.0.0.5: icmp_seq=1 ttl=64 time=1258 ms
64 bytes from 10.0.0.5: icmp_seq=2 ttl=64 time=257 ms
64 bytes from 10.0.0.5: icmp_seq=19 ttl=64 time=141 ms
64 bytes from 10.0.0.5: icmp_seq=20 ttl=64 time=90.2 ms
^C
--- 10.0.0.5 ping statistics ---
22 packets transmitted, 18 received, 18% packet loss, time 21052ms
rtt min/avg/max/mdev = 80.708/179.166/1258.342/264.490 ms, pipe 2
```

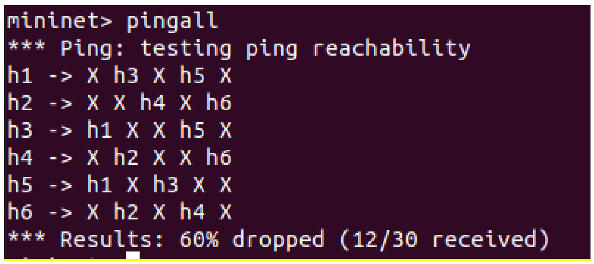
Figure 12: High latency and packet loss between H1 and H5 (After attack)



```
mininet>
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=35.0 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=38.2 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=31.1 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=45.7 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=31.7 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=47.0 ms
^C
--- 10.0.0.3 ping statistics ---
27 packets transmitted, 18 received, 33% packet loss, time 26081ms
rtt min/avg/max/mdev = 26.968/39.420/50.516/7.118 ms
mininet>
```

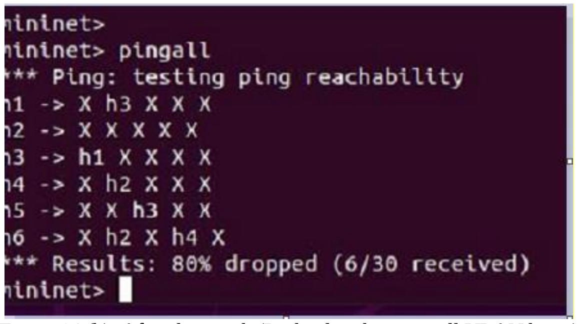
Figure 13: High latency and packet loss between H1 and H3 (After attack)

Before and after attack results are shown above in Figures 10, 11,12 and 13 proved that the attack on VLAN 300 impacted the reachability of hosts in VLAN 200. The final test of pingall showed the significant packet loss in the SDN environment with the host deployed in multiple VLANs. The same VLAN hosts can ping each other. The same can be seen in figure 14(a).



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X h5 X
h2 -> X X h4 X h6
h3 -> h1 X X h5 X
h4 -> X h2 X X h6
h5 -> h1 X h3 X X
h6 -> X h2 X h4 X
*** Results: 60% dropped (12/30 received)
```

Figure 14 (a): Before the attack (same VLAN hosts successfully ping each other)



```
mininet>
mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 X X X
h2 -> X X X X X
h3 -> h1 X X X X
h4 -> X h2 X X X
h5 -> X X h3 X X
h6 -> X h2 X h4 X
*** Results: 80% dropped (6/30 received)
mininet>
```

Figure 14 (b): After the attack (Packet loss between all VLAN hosts)

The final results of this experiment proved that the DDoS attack in one VLAN impacted the services of other VLAN. VLAN was created to segment the traffic with no intervlan communication but huge traffic generated through DDoS attack in one VLAN exhausted the resources of the controller, which delayed the response of legitimate traffic from other VLANs and results in Denial of Service for that VLAN.

IV. Conclusion

In this paper, experimental evaluation has been performed to measure the throughput and latency of the Software-defined Network Controller when it is under DDoS attacks.

This experiment was conducted in two modules, without any security implementation on the communication channel and with security implementation at the communication channel. In both cases, multiple switches were used to overwhelm the controller with multiple requests.

The results of the experiment showed that implementing a security layer at the communication layer and adding a VLAN between the controller and the data plane was able to detect and reduce the possibility of DDoS attacks. It was demonstrated by conducting various DDoS attacks such as TCP flood, UDP, and HTTP floods using the LOIC attack tool in a controlled environment. These floods at different paces overwhelmed the controller and impacted the processing of the requests at a greater rate. However, the controller with the secure communication layer performed better with higher throughput and lower latency as compared to the controller without it.

Overall, OpenDayLight Controller showed a lower throughput and higher latency when a DDoS attack was performed. However, it also showed that the SDN architecture can perform better in terms of confidentiality when it has a proper security method implementation such as SSL layer, VLAN, and VPN. In the future, the research may be extended to focus on more advanced security mechanisms for the SDN controller which can prevent DDoS attacks to a greater extent.

V. References

[1] I. TechTalk, *What is Software Defined Networking,* 2017.

[2] I. Services, "SDN versus Traditional Networking," 2019.

[3] Z. Latif, K. Sharif, F. Li, M. M. Karim and Y. Wang, "A Comprehensive Survey of Interface Protocols for Software Defined Networks," *Journal of Network and Computer Applications,* vol. 156, 2020.

[4] D. B. Hoang and M. Pham, "On Software defined Networking and the design of SDN Controllers".

[5] Admin, "Southbound vs Northbound SDN: What are the differences?," 17 02 2020. [Online]. Available: https://www.webwerks.in/blogs/southbound-vs-northbound-sdn-what-are-differences.

[6] "SDN Northbound interfaces (NBI) and Southbound interfaces (SBI)," February 2017. [Online]. Available: https://netfv.wordpress.com/2017/02/13/sdn-northbound-interfaces-nbi-and-southbound-interfaces-sbi/.

[7] O. Blial, M. B. Mamoun and R. Benaini, "An Overview of SDN Architectures with mulitple SDN Controller," *Journal of Computer Networks and Communications,* vol. 2016, no. 9396525, p. 8.

[8] N. A. Aziz, T. Mantoro, M. A. Khairudin and A. F. b. A. Murshid, "Software Defined Networking (SDN) and its Security Issues," *4th International Conference on Computing, Engineering, and Design (ICCED),* 2018.

[9] B. Lin, X. Z. Ding and Zhiguo, "Research on the Vulnerability of Software Defined Network,"

Advances in Engineering Research (AER), vol. 148, p. 8, 2017.

[10] N. Hoque, M. Bhuyan, H., R. Baishya, D. Bhattacharyya and J. Kalita, "Network attacks: taxonomy，tools and systems.," *Journal of Network and Computer Applications,* Vols. 307-324, p. 18, 2014.

[11] R. K. Arbettu, R. Khondoker, K. Bayarou and F. Weber, "Security Analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN Controllers," *IEEE,* pp. 39-42, 2016.

[12] "Opendaylight : Security Vulnerabilities," CVE, 2018. [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-13628/Opendaylight.html.

[13] M. Iqbal, F. Iqbal, F. Mohsin, D. M. Rizwan and D. F. Ahmad, "Security Issues in Software Defined Networking(SDN): Risks, Challenges and Potential Solutions," *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 10, 2019.

[14] A. Sebbar, M. Boulmalf, M. D. E.-C. E. Kettani and Y. Baddi, "Detection MITM Attack in Multi-SDN Controller," *IEEE 5th International Congress on Information Science and Technology (CiSt),* 2018.

[15] Q. Ilyas, "Security Analysis of FloodLight, ZeroSDN, Beacon and POX SDN Controllers," in *SDN and NFV Security,* 2018, pp. 85-98.

[16] J. Galeano-Brajones, j. Carmona-Murillo, J. F. Valenzuela-Valdés and F. Luna-Valero, "ncbi.nlm.nih.gov," MDPI, Basel, Switzerland, 3 2 2020. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7038683/.

[17] Y. E. Oktian, S. Lee, H. Lee and J. Lam, "Secure your Northbound SDN API," *IEEE,* vol. 10.1109/ICUFN.2015.7182679, p. 2, 10 August 2015.

[18] R. M. Thomas and D. James, "DDOS detection and denial using third party application in SDN," *IEEE,* vol. 10.1109/ICECDS.2017.8390193, p. 6, 2018.

[19] C. Banse and S. Rangarajan, "A Secure Northbound Interface for SDN Applications," *IEEE,* vol. 10.1109/Trustcom.2015.454, pp. 834-839, 2015.

[20] O. Polat and H. Polat, "The effects of DoS attacks on ODL and POX SDN controllers," *8th International Conference on Information Technology (ICIT),* 2017.

[21] N. S. Team, "What Is the Low Orbit Ion Cannon (LOIC)?," netsparker, 24 July 2019. [Online]. Available: https://www.netsparker.com/blog/web-security/low-orbit-ion-cannon/.

[22] A. Bidaj, "Security Testing SDN Controllers," *Aaltodoc "http://urn.fi/URN:NBN:fi:aalto-201608263040",* p. 6+61, 2016-07-29.

[23] B. Golden, Virtualization for Dummies, Wiley Publishing, Inc., 2007.

[24] K. Benzekki, A. E. Fergougui and A. E. Elalaoui, "Software-defined networking (SDN): a survey," *Wiley Online Library,* no. https://doi.org/10.1002/sec.1737, 04 03 2017.

[25] A. Shaghaghi, M. A. Kaafa, R. Buyya and S. Jha, "Software-Defined Network (SDN) Data Plane Security: Issues,," *Cluster Computing Journal,* p. 24, 2018.

[26] S. Scott-Hayward, G. O'Callaghan and S. Sezer, "SDN security: A survey," *IEEE,* 2013.

[27] "Open Networking Foundation Formed to Speed Network Innovation," Open Networking Foundation, 30 12 2020. [Online]. Available: https://web.archive.org/web/20110326024026/http://www.opennetworkingfoundation.org/?p=7. [Accessed 01 03 2021].

[28] M. Gozani, Network Virtualization For Dummies®, VMware Special Edition, John Wiley & Sons, Inc., 2016.

[29] "Network Functions Virtualisation (NFV)," ETSI, 16 02 2021. [Online]. Available: https://www.etsi.org/technologies/nfv. [Accessed 02 03 2021].

[30] P. Goransson, C. Black and T. Culver, "Genesis of SDN," 2017.

[31] Pradeepa.R and Pushpalatha.M, "Exploring Attack vectors and Security," *International Journal of Innovative Technology and Exploring Engineering (IJITEE),* vol. 8, no. 11, p. 5, 2019.

[32] S. M. Mousavi, "Early detection of DDoS attacks in software defined networks controller.," *IEEE,* no. Diss. Carleton University,, 2014..

[33] C. Kolias, G. Kambourakis, A. Stavrou and J. Voas, "DDoS in the IoT: Mirai and other botnets.," *Computer ,* vol. 50, pp. 80-84, 2017.

[34] "Brief History of Virtualization," Oracle.com, [Online]. Available: https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html. [Accessed 20 03 2021].

[35] S. Singh and S. K. V. Jayakumar, "A Study on Various Attacks and Detection Methodologies in Software Defined Networks," *https://doi.org/10.1007/s11277-020-07387-y,* vol. 114, no. 1, pp. 675-697, 2020/09/01.

[36] C. Banse and S. Rangarajan, "A Secure Northbound Interface for SDN Applications," *IEEE,* no. DOI 10.1109, pp. 834-839, 2015.

[37] J. Haasz, "802.1D-1990 - Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges," *IEEE,* 31 05 1990.