SOFTWARE DEFINED NETWORKING SECURITY

Authored by

Oluwatola Adeniyi (142726)

Digvijay Chauhan (141807)

Ugochukwu Ezidonye (142046)

Mojisola Ayeni (142495)

Vishwa Patel (141259)

Research Project

Submitted to the Faculty of Graduate Studies,

Concordia University of Edmonton

In Partial Fulfillment of the Requirements for the Final

Research Project ISSM581(D)

Concordia University of Edmonton

FACULTY OF GRADUATE STUDIES

Edmonton, Alberta

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

April 2021

Table of Contents

List of Tables

List of Figures

Software-Defined Networking Security

Oluwatola Adeniyi; Digvijay Chauhan; Ugochukwu Ezidonye; Mojisola Ayeni; Vishwa Patel
Department of Information Systems Security Management
Concordia University of Edmonton
Alberta , Canada
oadeniyi@student.concordia.ab.ca; dchauhan@student.concordia.ab.ca; uezidony@student.concordia.ab.ca; mayeni@student.concordia.ab.ca; vpatel1@student.concordia.ab.ca
Advisor: Dr Sergey Butakov; sergey.butakov@concordia.ab.ca

Abstract— The evolution of communication technologies led to active virtualization of computer network solutions and proliferation of the Software Defined Networks (SDN) as a centralized network infrastructure management tool. The SDN is a relatively new technology and rush to move products to the market in some cases forces companies to overlook security mechanisms. The proposed research provides assessment of the susceptibility of some of the SDN solutions to the DDoS attacks. The attacks on north- and south-bound communications have been carried out using two SDN controllers on a public cloud. Detailed analysis of the attack results indicated that even small-scale DDoS floods can have devastating impacts on the SDN ecosystem if the attacking botnet has access to communication channels that carry out network management traffic. The research makes it is evident that proper isolation is required, and it should not be limited to standard mechanisms as encryption but should assume proper authorization mechanisms to prevent such attacks.

Keywords—SDN, DDoS attack, Cbench, ODL, Ryu

I. INTRODUCTION

Networks have grown significantly in detail and size, which has made the movement of hardware switches a very difficult task. For companies that operate virtualized systems with large networks, simply installing device switches through a manual process has become a complicated and vulnerable task. Software Defined Networking (SDN) is a networking process that allows system administrators to control and run large networks more reliably. It accomplishes this by separating the various layers of a particular network, a process known as network abstraction. SDN offers many significant benefits for organizations, networks, and service providers by using a controller based computerized environment. ***[1] [2] [3]***. SDN has several advantages which include ***[4]:***

i. **Offers a Centralized Network:** It provides a standardized view of the overall network and simplifying business operations**.**
ii. **Offers a Broad and Extensive Management:** SDN allows IT employees, managers, or supervisors to test configuration settings without affecting the network.
iii. **Offers a Compact and Efficient Security:** it provides a centralized point of control for administering security and set of policies over the network. This could serve as a weakness because it makes the SDN controller a point of failure or target. If installed in a secure and correct manner, it can effectively manage security all around the network.
iv. **Lowers Operating Expenses:** it allows device improvement simple and straightforward. It lowers the operating expenses of any organization, particularly in the administrative area. This is because most of the administrative matters or issues that occur from normal routines can indeed be replaced by automation and centralized.
v. **Guarantees Content Delivery across the Network:** One of SDN's key advantages is its ability to monitor network traffic. The ability to manage and improve traffic flow in a network makes quality of service (QoS) and communication simple to incorporate.

II. SDN ARCHITECTURE AND IMPLEMENTATION

Unlike traditional networks, SDN is a better alternative because it does not rely on the use of specific network devices like firewalls, routers, switches etc. dedicated hardware devices such as routers, switches, firewall etc. and lack of traffic management in the network. [5]The SDN consists of following three layers:

- Infrastructure/Data Layer
- Control Layer
- Application Layer

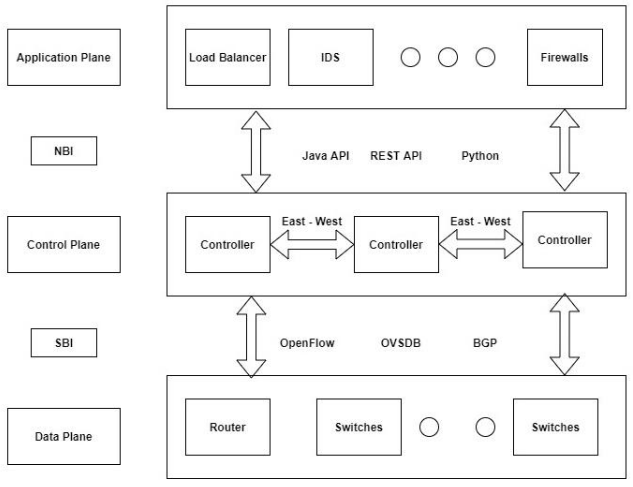


Figure 1: Basic SDN Architecture [6]

The SDN architecture is split into three different sections. The control layer is recognized as the brain of the network. It is also known as the centralized controller This same controller resides in a server that manages the network's traffic rules and connections. The infrastructure layer of a network is made up of physical switches. These layers interact and establish communication through the northbound and southbound application. [7] [8].

A. Application Layer

This layer can also be known as the application plane. All the services, features, and policies are defined in this plane. These applications are like programs that communicate and interacts its network requirements directly to the controller dynamically through the northbound interface. Depending on the network modifications, these applications can generate end-to-end functionalities.

B. Control Layer

This layer can also be known as the Control Plane. This layer offers a systematic centralized method of control that regulates network transmission operation through an accessible user interface. The SDN controller manages all or most of the devices that make up the network and enforces and deploys policies such as packet forwarding, routing, and load balancing via the southbound interface [9]. The Control layer is the most essential part of the SDN layers.

C. Infrastructure/Data Layer

This layer can also be known as the data plane. All the network devices are physically assembled and connected at this layer. Its forwarding operation can be executed based on the application layers policies or rules and actions from the controller. This layer's devices are just forwarding devices with no control or standard functions. The flow table in the network oversees handling of each packet. [10]

D. Northbound Interface (NBI)

Applications communicate with the controller via this interface as this can be shown in Figure 1. It oversees communicating network requirements and behavioral patterns from the application layer, as well as the SDN controller's view. The commands used to configure the forwarding devices are abstracted in the northbound interface. [11].

E. Southbound Interface (SBI)

This configurable interface is used by the controller to dynamically control the network devices in the data layer. The defined protocol for this interface is called OpenFlow. All forwarding functions, alerts, tracking, and can be controlled programmatically. Between both the data plane and the SDN controller, OpenFlow creates a safe channel. [11].

For administration of rules or policies across an organization or network, the SDN does have a centralized control point which can also be a major drawback, as it makes the SDN vulnerable to an attack. The SDN is a new technology, and it is susceptible to attacks because of lack of skilled personnel who might not understand the new technology. It consists of new codes, thereby increasing the vulnerability of the network. Most of the attacks that will be carried out will be done against the NBI and SBI. This is to prevent applications in the NBI from communicating with the controller and network devices in the SBI from communicating with the controller. Further details on steps that will be carried out to perform these attacks will be explained in the next section.

III. REVIEW OF RELATED WORKS

Several strategies have been used to assess the safety of SDN controllers as a subset of the overall SDN architecture. Using the STRIDE threat modeling, these assessments can be better classified. Microsoft created the STRIDE threat modeling system. Praerit and Loren oversaw identifying and classifying computer security threats. It divides threats into six categories as shown below[12].

Table 1: Stride Threat Modelling [12]

Threat	Definition	Property
Spoofing	Imitate something or someone else	Authentication
Tampering	Interfering with data	Integrity
Repudiation	Pretending to have completed a procedure	Non-repudiation
Information Disclosure	Information being given to someone who isn't supposed to have it	Confidentiality
Denial of Service	Users are being denied service.	Availability
Elevation of privilege	Obtaining access without having appropriate authorization	Authorization

The threats discussed above can be performed against the controller either directly or via the northbound or southbound interface. As a result of this, securing the controller and the communication channels is key to having a safe network. Below are some of the techniques attackers may use to perform these threats.

i. **Spoofing** – An attacker may forward packets with a source address to the SDN controller showing that the packet is originating from a port or device in the network. The attacker can gain access to the network through ARP spoofing.

ii. **Tampering** – if the network is not properly configured, an attacker can intercept packets sent from the controller to the NBI. These packets can be modified or redirected thereby giving the malicious actor access to modify policies on the flow table.

iii. **Repudiation** – Since all activities carried out by system administrators are logged. If there is a malfunction in the logs or tracking systems, interactors in the system can deny their actions or even blaming others if anything goes wrong.

iv. **Information disclosure** – an attacker may gain unauthorized access to sensitive information in the network such as the topology, configuration details, flow table or the cryptography key. The cryptography key is vital because it can be used to verify the controller's identity.

v. **Denial of Service - an** attacker may decide to forward a huge amount of traffic to flood the NBI or controller in the network thereby making the NBI and controller unavailable.

vi. **Elevation of Privilege –** If there is a vulnerability with the access control in the system, a malicious user can escalate his privilege in the system which will then enable unconfined access to restricted or sensitive data.

Table 2 below shows different proposed mitigation techniques from different authors showing countermeasures to tackle the spoofing, Man-in-the-Middle attack, and denial of service attacks.

Table 2: Mitigation Techniques for SDN Attacks

Attacks	**Impacted SDN Layers and Interface**	**Affected Security Aspect**	**Mitigation Techniques**
DOS/DDoS	Control, Infrastructure, SBI	Availability	• Statesec [13] • Openflow Switch [14] • Sguard [15] • Avantguard [16]
Spoofing	Control, Infrastructure	Confidentiality, Integrity	• ARP Spoofing Mitigation [17] • SDSec [10] • Hybrid SDN [18]
Man-in-the-middle	Control	Availability, Integrity	• Snort [19] • ArpAlert [19]

Boite et. al. proposed using the Statesec. This is a security management tool used for the detection of DoS attack. It functions in three stages which are: Irregularities detection, traffic management and remediation countermeasures. This uses a stateful technique to employ switch processing, that can aid in the precise detection and mitigation of DoS attacks. It also aids by lowering the amount of connection overhead that occurs in the south-bound interface. [13] Huang et. al. proposed using the OpenFlow keeps track of the flow of traffic These data can be regulated to see if there is a substantial increase in traffic, which could indicate a DDoS attack. [14] Wang et. al. proposed using the Sguard which is a powerful security software for detecting DoS attacks. It is divided into two parts: classification and access control. The Access control keeps a record of a packet's actual source and matches it up to the Hash table as it is entering a network. [15] Shin et. al. proposed using the Avantguard. This issue is fixed with the support of a migration module which slows down the TCP handshake by allowing the switch serve as a proxy server, thereby forwarding the completed connections to the controller. This technique can help to prevent an attacker from completing the TCP handshake. [16] Abdelsalam et. al. proposed using the ARP Spoofing Mitigation. This method involves monitoring port level ARP packets by adding an ARP function in the control unit for effective spoofing detection and prevention. It can also protect a control system from overloading in the event of an attack. [17] Darabseh et. al. proposed using the SDSec. This technique is based on an open switch controller. To identify the active connections in the network, it employs the Link Layer Discovery Protocol (LLDP). [20] Fahad et. al. proposed using the Hybrid SDN. This technique involves creating a new server dedicated to receiving all ARP queries. By redirecting malicious activity from attackers to the newly created server. [18] The use of tools like Snort and ArpAlert can help to tackle the Man-in-the-middle attack. These tools can be installed and used to detect and prevent attacks from occurring. This would involve setting up and configuring these tools on an interface in the same network as the SDN controller. [19]

The authors above proposed different countermeasures to effectively mitigate some of the security issues affecting the SDN. Some of these analysis and assessment did not give a detailed and comprehensive solution to some issues affecting the northbound communication. Most of the proposed solutions could handle simple DoS attacks but whenever the attacks became stronger, detecting was possible but identifying the attacker was not easy. Another common issue was detecting slow DDoS attacks. This is because the slow DDoS attacks attempts to completely slip through undetected. The idea behind this research proposal is to tackle these issues and propose possible solutions to better harden the network and make it less prone to attacks.

IV. METHODOLOGY

The Software defined network was configured on a public infrastructure with two opensource controllers and SDN benchmarking tools to conduct the experiment. The benchmarking tools used are cbench , apache bench (ab) tool, and iperf. [21]

A. Experiment Setup

For this experiment, eight (8) virtual machines running ubuntu 20.0.4 were used to represent the application layer, control layer, data layer, and external attackers. The two controllers used were Opendaylight controller running on the sodium version, and Ryu controller.
The first machine (3 vCPUs, 40 GB HDD, 4GB RAM) serving as the application layer had the apache bench installed to measure the impact of a DDoS attack on the northbound communication. The second virtual machine (3 vCPUs, 40 GB HDD, 4GB RAM) running Opendaylight sodium with all required features installed (DLUX and L2), and Ryu controller. The third VM running Mininet, (2 vCPUs, 40 GB HDD, 4GB RAM) used to the emulate the data layer, and cbench the performance monitoring tool used to measure the impact on

southbound communication. The other five (5) virtual machines served as external attackers with hping3 and dsniff package installed to perform DDoS and man-in-the-middle attacks, respectively.

For attack at the data layer, Mininet was used to emulate the data layer topology with seven (7) switches and eight (8) hosts. And for attacks against the controller, the data layer topology was emulated using cbench, where 16 switches and 1000 hosts were tested to ascertain what the impact of the DDoS attack on the controller would be on the southbound communication channel.

Since the northbound communication uses REST APIs to retrieve the switch stats and update the switch stats for Ryu controller, and get information on the network, flow statistics, and host locations for ODL, Apache Bench [22]a benchmarking and load-testing tool, was used to confirm how the controller would behave when under attack. The tool uses http to send requests on port 8080 of the Ryu controller and 8181 for the ODL controller.

Cbench [21] is a benchmarking tool for controllers that support OpenFlow 1.0 and 1.3. It supports two metrics i.e., throughput and latency. Where cbench measures latency by forwarding a single packet-in data to the controller and awaiting a response, while for throughput, it repeatedly forwards as many packet-in data as possible to measure the controller's capability. Tests for these two metrics can be done multiple times.

B. Experiment Scenario

i. Attack at the data layer

For the first attack, a distributed denial of service attack was simulated where the attacker is a host in the same network as the victim in the data layer. The type of DDoS attack used in this scenario is syn flood attack where h1 sends SYN packets to h8. Prior to the attack, the bandwidth of the victim needed to be verified and this was done using the iperf utility tool. Using the iperf command, the bandwidth was at 10Mbit/s when a connection was established from a normal user, h2 to the victim h8. To perform the attack, the hping3 tool was used to perform a SYN flood attack. Once the attack begins from h1, the bandwidth reduces drastically from 10Mbits/s to an average of 250 Kbits/s.

ii. Attack at the Southbound Interface

a. Distributed Denial of Service Attack

The southbound interface is responsible for controller-data layer communication. An attack against the controller would impact the request and response times for openflow packets sent from the openflow switches to the controller, flowmod or packet out messages from the controller to the switches.

To perform an effective DDoS, five attackers were used at various attack speeds. To configure the attack with these intervals, the hping3 tool was used, these tools allow for the speed of DDoS to be configured by setting the *-i value*, and in this case, it was calibrated from the slowest to fastest. This means the interval or speed value was set to *u1000* to indicate slow attack, that is, the attacker would send approximately 100 packets per second to the victim, while a value of *u10* would be a very fast attack. For the purpose of this attack, these intervals were calibrated as follows: *u1000, u750, 500, 300. 200, 150, 100, 50, 25, 10.*

Cbench was used to measure the impact of the DDoS attack at these intervals on the throughput on the southbound interface. It was configured with 16 switches and 1000 mac addresses per switch. 110 iterations were executed per switch at 10000 ms per test. The first 10 iterations and loops are ignored from the results.. This was done to cater for any variance in the results. The results from using cbench are an average of the number of responses per second from all switches against the speed or intensity of the attack.

It is seen that both controllers can withstand the attack from one attacker but when the number of attackers and speed of attack are increased, the average responses per second decreases, that is, how long it takes the controller to respond to a packet_in message sent by the cbench tool.

From figures 2 and 3 , the average response per second when the DDoS is executed on the ODL controller using 5 attackers is 26.58 and this will decrease further as the speed of attack and number of attackers increases, while for the Ryu controller the average response per second is 135 during attack with high intensity using five attackers.With one attacker, this result varies a little, as the controller is able to withstand the attack even if the speed increases, hence the increase in the response per second. From the graphs, the variation can be noticed in both controllers and this is due to how both controllers handle DDoS, ODL handles this fairly as the responses per second is more than the Ryu, even with five attackers.

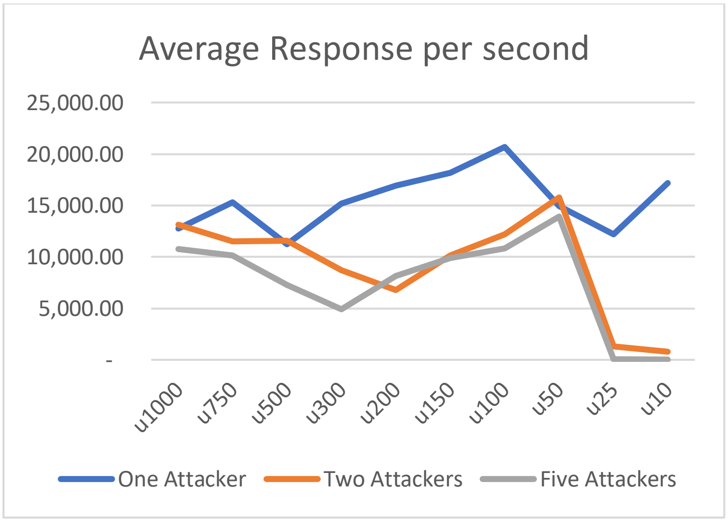


Figure 2: Average responses per second in ODL for one, two and five attackers

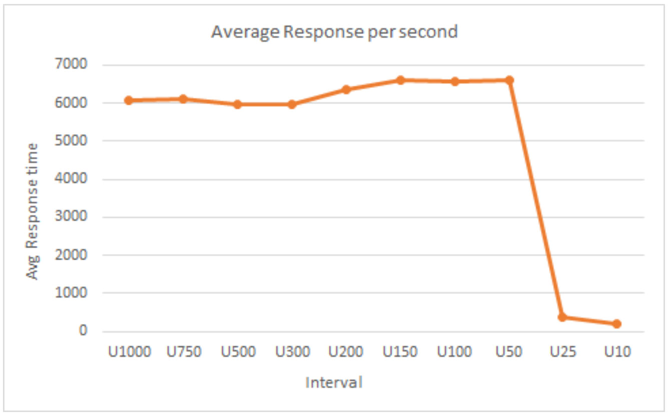
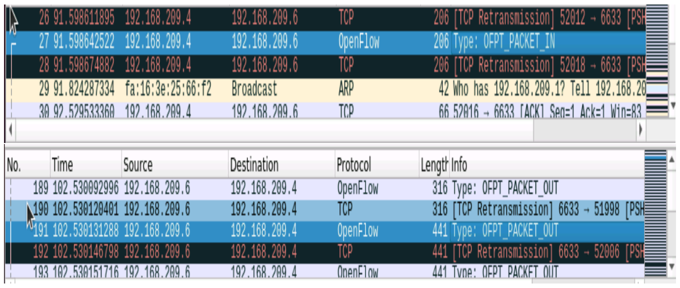


Figure 3: Average responses per second in Ryu using 5 attackers

b. Man- in-the-Middle Attack.

To intercept SBI channel communication, man-in-the-middle attack was performed by poisoning the ARP cache of the controller VM. The attacker intercepts the exchange of packet-in and packet-out messages between the controller and the OpenFlow switches and uses this information to get an overview of the topology. It can take it a step further by modifying the flow of information sent to the controller which could lead to rerouting information or traffic to a compromised host. Figure 4 shows the Wireshark capture of this communication between the switches and ODL controller. Here, 192.168.209.6 is the controller VM and 192.168.209.4 is the mininet VM.



```
 26 91.598611895  192.168.209.4   192.168.209.6   TCP       206 [TCP Retransmission] 52012 → 6633 [PSH
 27 91.598642522  192.168.209.4   192.168.209.6   OpenFlow  206 Type: OFPT_PACKET_IN
 28 91.598674882  192.168.209.4   192.168.209.6   TCP       206 [TCP Retransmission] 52018 → 6633 [PSH
 29 91.824287334  fa:16:3e:25:66:f2 Broadcast     ARP        42 Who has 192.168.209.1? Tell 192.168.20
 30 92.529533360  192.168.209.4   192.168.209.6   TCP        66 52016 → 6633 [ACK] Seq=1 Ack=1 Win=83

No.  Time          Source          Destination     Protocol  Length Info
 189 102.530092996 192.168.209.6   192.168.209.4   OpenFlow  316 Type: OFPT_PACKET_OUT
 190 102.530120401 192.168.209.6   192.168.209.4   TCP       316 [TCP Retransmission] 6633 → 51998 [PSH
 191 102.530131288 192.168.209.6   192.168.209.4   OpenFlow  441 Type: OFPT_PACKET_OUT
 192 102.530146798 192.168.209.6   192.168.209.4   TCP       441 [TCP Retransmission] 6633 → 52006 [PSH
 193 102.530151716 192.168.209.6   192.168.209.4   OpenFlow  441 Type: OFPT_PACKET_OUT
```

Figure 4: Wireshark capture of this communication between the switches

iii. Attack at the Northbound Interface

a. Distributed Denial of Service Attack

Two attackers were used to perform a DDoS attack on the controller and apache bench tool was used to measure the impact on the application-controller communication. The parameter used to analyze the behavior was the request per time value from apache bench. Prior to the attack, this average requests per second for ODL was approximately 1185 while for Ryu the average requests per second was 677. With similar intervals as mentioned in b above, the number of connections were set to 100 with a concurrency level of 10. For the two attackers, the average request per time for ODL and Ryu are seen in figures 5 and 6.

For both controllers, during the attack, the time taken to process one request from the application layer increases slightly as the speed of the attack increases, when the attack is executed with one attacker. The only significant change is at *u25* and *u10*, where it takes the controller a longer time to process a single request. When the number of attackers is increased, there is a major rise at the fastest point, and this is seen in figures 5 and 6.

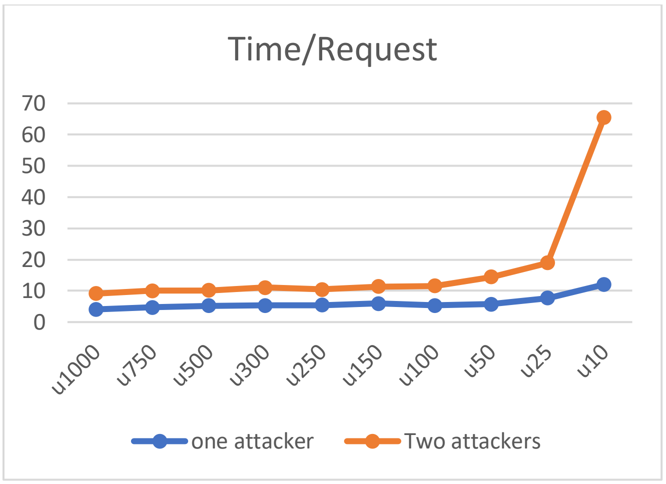


Figure 5: Time/Request for ODL using two attackers

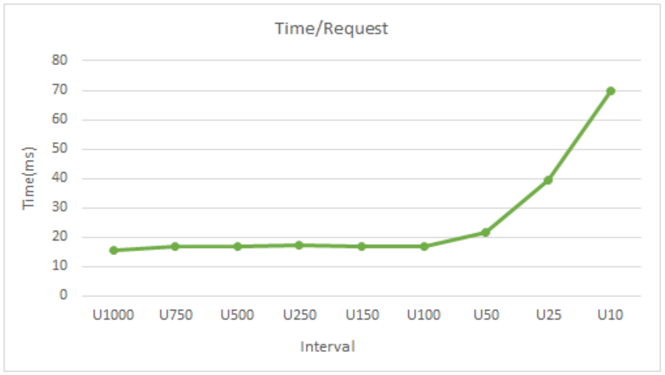


Figure 6: Time/Request for Ryu using two attackers.

The number of requests each controller can handle declines as the intensity and number of attackers increases. In figure 7, the number of requests ODL processes per second decreases from approximately 2000 requests/second to almost 190 requests/second. This indicates that at high intensity attack and a high number of attackers, ODL will not be able to withstand the DDoS attack.

A similar observation is made for the Ryu controller in figure 7, Ryu can handle an average of 650 requests per second when the attack intensity is low. As the intensity increases, it is seen that Ryu struggles to handle the number of request it receives.

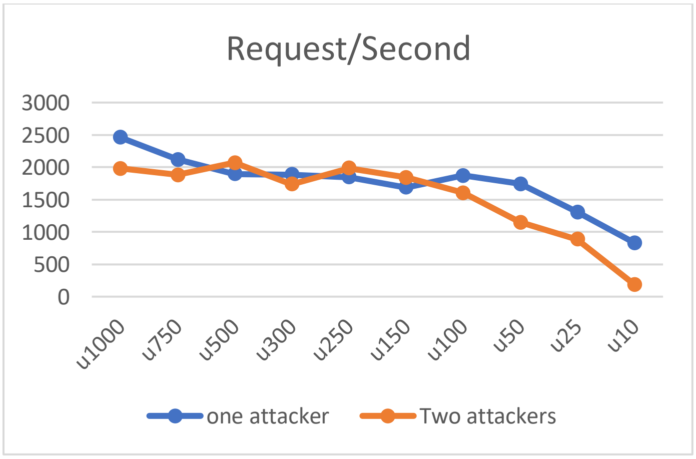


Figure 7: Average requests per second for ODL using two attackers

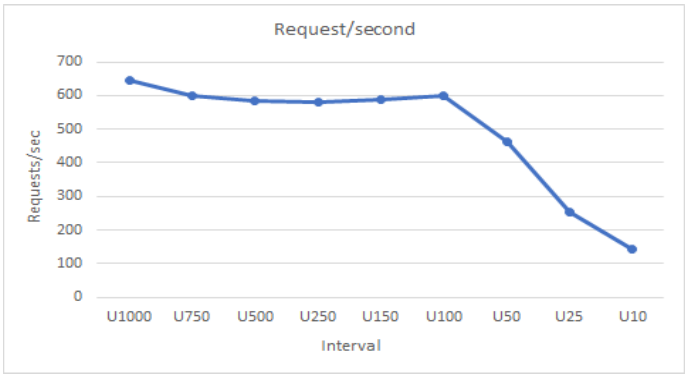


Figure 8: Average requests per second for Ryu using two attackers

In addition to the experiments above, DDoS attack was executed against the controller while a VPN was configured between the control plane and data plane, to identify if this control will prevent the attack. 5 attackers were used at the same time with the maximum intensity reduced from the previous experiments.

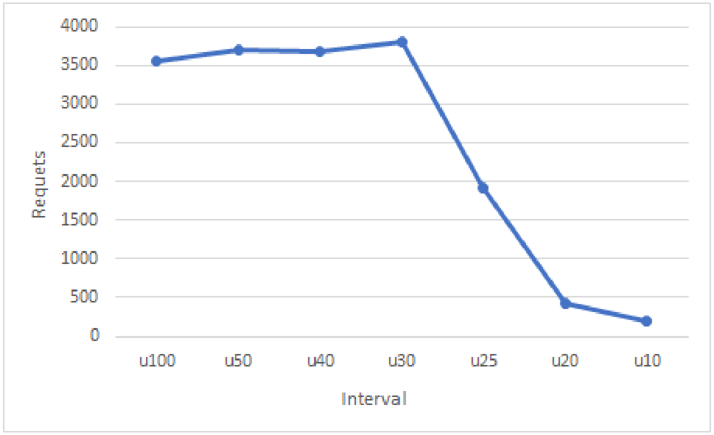


Figure 9: Result of DDoS with VPN using 5 attackers

It was observed that despite the presence of the VPN, it did not prevent the DDOS from impacting the infrastructure negatively. Here, the throughput when measured with cbench decreased drastically while DDoS was executed. And a similar result to figure 9 above was gotten. Therefore, it can be said that despite the ability of the VPN to mitigate man-in-the-middle attack, it cannot prevent DDoS attacks.

V. CONCLUSION

This paper assessed the susceptibility of SDN infrastructure to various forms of attack and how they impact confidentiality, integrity, and availability in software defined networks. It examined the effect on communication between the various layers, with attacks such as, distributed denial of service, and man-in-the middle executed at the various layers. From the analysis presented above, it is seen that DDoS floods have a tremendous impact on both the Northbound and Southbound communication channels.

With respect to the small scale DDoS attack executed, the results from the evaluation, using the benchmark tools (cbench, apache benchmark, iperf), show that both Opendaylight and Ryu controllers would not withstand an attack if the intensity of the attack was increased significantly, by increasing the number of attackers in the botnet. This is seen in the decline in the number of responses to requests sent from the application and data layers to the controller.

As there is no silver bullet solution to mitigate against any attack, future research can be done in analysing the integrity and confidentiality of the communication channels with the use of different VPN or other encryption methods, as well as using some virtual network isolation approaches as VLANs under different penetration techniques. Also, seeing as this experiment was performed at low scale with a single controller, more experiments should be done with a distributed arrangement for the controllers to test a fault tolerance system that ensures availability of SDN infrastructure.

VI. REFERENCES

[1] M. Cooney, "Network World," 2020. [Online]. Available: https://www.networkworld.com/article/3209131/what-sdn-is-and-where-its-going.html. [Accessed 23 May 2020].

[2] Q. Monnet, "Qmonnet.github.io," 8 July 2016. [Online]. Available: https://qmonnet.github.io/whirl-offload/2016/07/08/introduction-to-sdn/. [Accessed 23 May 2020].

[3] M. Rouse, "denial-of-service attack," September 2018. [Online]. Available: https://searchsecurity.techtarget.com/definition/denial-of-service. [Accessed 5 June 2020].

[4] MICROINGRAM, 08 August 2017. [Online]. Available: https://imaginenext.ingrammicro.com/data-center/7-advantages-of-software-defined-networking. [Accessed 05 06 2020].

[5] Vmware, 2018. [Online]. Available: https://www.vmware.com/topics/glossary/content/software-defined-networking.

[6] M. Rouse, "SearchNetworking," August 2019. [Online]. Available: https://searchnetworking.techtarget.com/definition/software-defined-networking-SDN. [Accessed 23 May 2020].

[7] D. Kreutz, R. M. V. Fernando, P. E. Verı´ssimo, C. E. Rothenberg and S. Azodolmolky, "Software-Defined Networking:A Comprehensive Survey," *Proceedings of the IEEE,* vol. 103, p. 31, 2015.

[8] L. Yifan, Z. Bo, P. Zhao, P. Fan and H. Liu, "A survey: Typical security issues of software-defined networking," vol. 16, no. 7, pp. 13-31, July, 2019.

[9] D. B. Hoang, "Software Defined Networking ? Shaping up for the next disruptive step?," 2015. [Online]. Available: http://doi.org/10.18080/ajtde.v3n4.28.. [Accessed 01 06 2020].

[10] J. Manar, T. Singh, A. Shami, R. Asal and Y. Li, "Software-Defined Networking: State of the Art and Research Challenges," Computer Networks, June 2014.

[11] M. Fakoorrad, "Application Layer of Software Defined," Tallinn, 2016.

[12] . L. Kohnfelder and . P. Garg, "The Threat to our Products," Microsoft, 1 April 1999. [Online]. Available: https://adam.shostack.org/microsoft/The-Threats-To-Our-Products.docx. [Accessed 5 June 2020].

[13] J. Boite, P.-A. Nardin, F. Rebecchi, M. Bouet and V. Conan, "Statesec: Stateful monitoring for DDoS protection in software defined networks," *2017 IEEE Conference on Network Softwarization (NetSoft),* pp. 1-9, 2017.

[14] X. Huang, X. Du and B. Song, "An effective DDoS defense scheme for SDN," *IEEE International Conference on Communications (ICC),* no. 978-1-4673-8999-0, 2017.

[15] T. Wang and H. Chen, "SGuard: A Lightweight SDN Safe-Guard Architecture for DoS Attacks," vol. 14, no. 6, pp. 113-125, 2017.

[16] S. Shin, V. Yegneswaran, P. Porras and G. Gu, "AVANT-GUARD: Scalable and Vigilant Switch Flow," ACM SIGSAC, November 2013.

[17] A. Abdelsalam and El-Sisi, "Mitigating ARP spoofing attacks in software-defined networks," pp. 25-27, 2016.

[18] F. Ubaid, R. Amin, F. B. Ubaid and M. M. Iqbal, "Mitigating Address Spoofing Attacks in Hybrid SDN," *International Journal of Advanced Computer Science and Applications,* vol. 8, no. 4, 2017.

[19] M. Brooks and B. Yang, "A Man-in-the-Middle Attack Against OpenDayLight," Proceedings of the 4th Annual ACM Conference on Research in Information Technology, September 2015.

[20] A. Darabseh, M. Al-Ayyoub, Y. Jararweh, E. Benkhelifa, M. Vouk and A. Rindos, "SDSecurity: A Software Defined Security experimental framework," *2015 IEEE International Conference on Communication Workshop (ICCW),* no. 2164-7038, 2015.

[21] R. Sherwood, "Github," [Online]. Available: https://github.com/mininet/oflops/tree/master/cbench. [Accessed 20 03 2021].

[22] D. Singer, "How to Benchmark a Website Using Apache Bench," Liquidweb, 18 November 2020. [Online]. Available: https://www.liquidweb.com/kb/how-to-benchmark-a-website-using-apache-bench/. [Accessed 20 March 2021].

[23] G. V. Nikolaev, "Network Monitoring with Software Defined networking Towards: OpenFlow network monitoring," 2013.