EMITTER International Journal of Engineering Technology
Vol. 9, No. 2, December 2021, pp. 252~267
DOI : 10.24003/emitter.v9i2.616

SDN-Based Network Intrusion Detection as DDoS defense system for Virtualization Environment

Saifudin Usman, Idris Winarno, Amang Sudarsono

Departement of Information and Computer Engineering,
Politeknik Elektronik Negeri Surabaya
Jl. Raya ITS Sukolilo Surabaya 60111, Indonesia
E-mail: saifudinu@politap.ac.id, amang@pens.ac.id
Correspondence Author: idris@pens.ac.id

Received September 10, 2021; Revised October 12, 2021; Accepted November 13, 2021

Abstract

Nowadays, DDoS attacks are often aimed at cloud computing environments, as more people use virtualization servers. With so many Nodes and distributed services, it will be challenging to rely solely on conventional networks to control and monitor intrusions. We design and deploy DDoS attack defense systems in virtualization environments based on Software-defined Networking (SDN) by combining signature-based Network Intrusion Detection Systems (NIDS) and sampled flow (sFlow). These techniques are practically tested and evaluated on the Proxmox production Virtualization Environment testbed, adding High Availability capabilities to the Controller. The evaluation results show that it promptly detects several types of DDoS attacks and mitigates their negative impact on network performance. Moreover, it also shows good results on Quality of Service (QoS) parameters such as average packet loss about 0 %, average latency about 0.8 ms, and average bitrate about 860 Mbit/s.

Keywords: DDoS, High Availability, Cloud Computing, Virtualization, NIDS, SDN, Sflow, Openflow

1. INTRODUCTION

Virtualization is the key to cloud computing technology, where many industries widely use it today. Sharing a single physical computer between multiple isolated virtual machines leads to more optimized hardware use, allowing virtual device relocation and maintenance more effectively than its physical equivalent. Virtualization is used at different levels, such as networks, CPU, memory, storage, etc. It improves system availability and thus lowers costs, offering a superior scalable system [1]. Although virtualization offers many advantages, it brings new security challenges; the hypervisor's implementation introduces threats as hypervisors expose new attack vectors. One of the most significant security attacks that threaten service availability in the cloud or virtualization environment is Distributed Denial of Service (DDoS) attack [2].

In a cloud virtualization environment, DDoS attacks are aimed at virtual servers or instances, as multiple users may experience flooding even on the same cloud server. As workloads increase, cloud systems provide more computing power by engaging more virtual machines or service instances, and eventually, cloud services slow down, and legitimate customers lose access to their cloud services. DDoS can significantly reduce cloud services' performance by damaging instances or virtual servers in the cloud environment. DDoS attacks may be much more severe if attackers use more zombie machines to attack many systems. Some of the most complex DDoS attacks targeting Cloud Computing based on "2020 DDoS Attack Landscape" are UDP Flooding Attack, SYN Flooding Attack, ACK Flood, HTTP Flood, SSDP Reflection Flood, NTP Reflection Flood, ICMP Flooding Attack, DNS Reflection Flood, and DNS Request Flood Attack [3]. Thus, a network-based security strategy is needed. Software-Defined Networks (SDN) become alluring in the fight against network-based attacks because SDN can easily collect network usage information, supporting improved algorithm designs to detect network-based attacks.

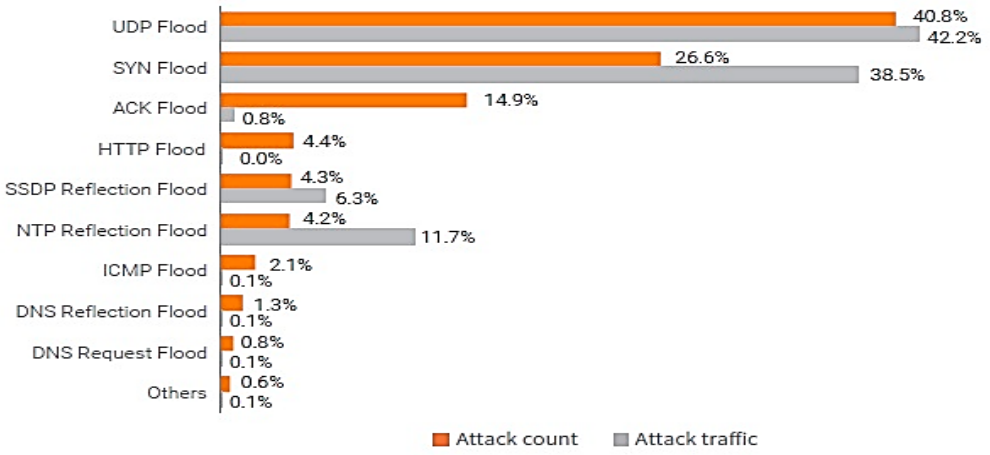

Figure 1. Proportions of Different DDoS Attack Types [3]

SDN is a modern paradigm of architecture, deployment, and management of networks. This new generation of digital networks is built to make networks more scalable, agile, and dynamic to meet today's rapidly evolving business needs. Mobile device's rapid growth and emerging technologies such as the Internet of Things, Cloud Computing, and Virtualization Environment are some of the developments affecting SDN progress and at the field of modern networking has become a rising trend if as compared to conventional networks, this is due to its design flexibility. While the invention of SDN allows a network device dynamic, there are also some risks because it is generally susceptible to a single point of failure (SPOF), traffic diversion, side-channel attack, network manipulation, exploitation of Application Programming Interface (API), traffic sniffing, distributed denial of service attack and Distributed Denial of Service (DDoS) attack. In short, the

attack depletes the network's capacity by overloading it with immense traffic at a time, rendering it inaccessible to legitimate users.

2. RELATED WORKS

The study of DDoS attacks is well studied, and many approaches have been researched over the past few years to prevent them, but most are conventional target networks. In parallel, SDNs are starting to emerge, gaining increasing attention from many network players. We are considering, among other things, offering an SDN-based strategy to thwart DDoS attacks immediately. There are already many solutions to combat DDoS attacks, but this research study focuses on SDN-based infrastructure security issues. Based on observed, can be categorized two different security strategies for evaluating such solutions: Signature-based and Anomaly-based detection. Signature-based techniques are used against known types of attacks in which incoming traffic patterns are compared with available attack signatures contained in the knowledge base. Manso et al. [4] presented SDN-based IDS to detect and mitigate DDoS attacks from botnets or their sources. Badotra et al. [5] offered SNORT-based for early DDoS detection using Opendaylight and open networking OS. Campos et al. [6] deployed On-the-Fly monitoring and treatment of security events using real SDN/OpenFlow Switch. Authors Po-Wen Chi et al. [7] deployed AMI as a threat detection mechanism based on the SDN network using IDS integration. Yazdinejadna et al. [8] presented a kangaroo-based intrusion detection system (KIDS) on software-defined networks, uses consecutive jumps like a kangaroo for announcing the attacks both to the SDN controller and other IDSs.

Also, anomaly-based techniques have become popular in recent times, and they compare the incoming traffic pattern with the “normal” traffic pattern over a predefined period. There are several techniques offered here, including machine learning, statistical and artificial intelligence. For example, the authors of Lopez et al. [9] presented an intrusion detection and prevention system based on a Broflow traffic analyzer. The authors of Ombase et al. [10] presented Dos attack mitigation using rules-based and anomaly-based techniques using Bro IDS. And some authors using statistical anomaly-based detection techniques [11]–[14].

3. ORIGINALITY

Many studies have contributed to detecting and mitigating DDoS attacks, with various techniques or strategies proposed. In previous research study [15], has been developed an SDN-based network intrusion detection system to detect and mitigate DDoS attacks at the application layer, which have successfully implemented on virtualization servers against HTTP DoS attacks. Next, the research study were continued to detect and combat DDoS attacks running in the Proxmox-VE virtualization cluster environment. This research study goal is to detect and mitigate DDoS attacks while maintaining Quality of Service (QoS) legitimate users even when online services are under attack.

Additionally, the solutions are scalable and still open enough to improvise to accommodate the detection and mitigation of other types of DDoS attacks.

Applying a direct network traffic pattern analysis mechanism to the SDN controller will overload the centralized control area and force the SDN controller to downgrade its performance for these other tasks [16]. It is, likewise, introducing NIDS into an SDN network that requires implementing Port mirroring or Network TAP results in significant resource consumption, especially in high-traffic network environments. In contrast to previous studies presented in table 1, which uses Network Tap as packet-oriented monitoring of SDN-based intrusion detection systems as a mechanism for early detection and prevention of DDoS attacks [3-5], [8]. TAP mode scenarios require dedicated physical NIC devices, neither efficient nor flexible, to deploy in multi-node virtualization environments. With sFlow as a more flexible and lightweight agent for monitoring network traffic, which has tested and evaluated in a testbed of the production virtualization environment of Proxmox-VE. Centralized use of NIDS using the sflow agent on the SDN platform as an effective and practical strategy applied to virtualization environments to detect and combat DDoS attacks without adding more compute load to server nodes and avoiding network or communication overheads that are less affected by monitoring activities.

Table 1. Comparison with existing research

Publication	Problem approached	SDN based DDoS detection techniques	Capabilities
Song W, et al. 2018 [11]	Prevent DoS attack for both control and data planes in SDN	Statistic (Entropy)	Centralized Control
I Gde Dharma et al. 2015 [12]	DDoS Detection and Mitigation for SDN Controller	Statistic (Time-based pattern)	Centralized Control
Ombase, P.M. et al., 2017 [10]	DoS Attack Mitigation using Anomaly-based Techniques	Anomaly-based	Centralized Control
Campos M, et al., 2017 [6]	On-the-fly Monitoring and Treatment of security events	Signature-based (Network-TAP)	Centralized Control
Manso P, et al., 2019 [4]	DDoS attack detection and mitigation from botnet	IDS Integration (Network-TAP)	Centralized Control
Badotra S, et al., 2020 [5]	DDoS attack detection and mitigation for open networking OS	IDS Integration (Network-TAP)	Centralized Control
Yazdinejadna, et al.,2021 [8]	SDN-based architecture for attack detection and malicious behaviors	Flow-based and packet-based intrusion detection	Distiributed Control
In this research	DDoS attack on Virtualization Environment	Hybrid (NIDS + sFlow)	HA Controller, Flexible and Scalable with sFlow

As a contribution to this research study is a hybrid system that combines NIDS and sFlow, which is applied to a virtualization environment with multisite Nodes or clusters that support high availability systems. With the SDN-based hybrid NIDS and sFlow system, have succeeded in implementing the DDoS defense system without burdening or reducing performance on the Controller because the network traffic pattern analysis mechanism is not carried out by the Controller but assigned to the NIDS. In addition, by eliminating Network-TAP with sFlow, this system is more efficient in terms of network resource consumption, more flexible, and scalable to implement.

4. SYSTEM DESIGN

This section discusses the design and implementation of SDN-based network intrusion detection as a system capable of detecting and combating DDoS attacks in a virtualized environment. The solution also maintains Quality of Service (QoS) levels and prevents Single Point of Failure (SPOF) on controllers caused by a failure on the hardware side or network problems. The experimental infrastructure used to conduct this research is the Proxmox Production Virtualization Environment. This Proxmox Cluster Virtualization Environment consists of three Proxmox VE v6.3-3 nodes running on a physical computer with the following technical specifications:

- CPU Specification, 24 x Intel(R)Xeon(R) CPU E5-2620v2 @2.10Ghz
- RAM 64 GiB and SSD 512GiB.

Each node is also equipped with two network interfaces, where one NIC is for internet connectivity, and the other NIC is used for internal connectivity in a virtualized environment; the specifications run on each VM can also be seen in table 2. Each node also activates the Openflow Switch or OpenvSwitch mode connected to the SDN controller. In this study, Ryu is used as the Controller, which runs in a Container on one of the nodes. This SDN controller is also supported with High Availability and automatic Failover by utilizing the HA feature on the Proxmox-VE Clusters and Keepalived.

Table 2. Host specification on virtualization environment

Host	Application and tools	Specifications
Controller (C1, C2)	Ryu Controller, Keepalived	LXC; Ubuntu 20.04-64bit, 1GiB RAM, 2Core,8G Ceph Storage
NIDS	Snort, Sflow-RT, Pigrelay	VM; Ubuntu 20.04-64bit, 4GiB RAM, 4Core, 32G Ceph Storage
Target's Server	Apache2, Iperf	VM; Ubuntu 20.04-64bit, 4GiB RAM, 4Core, 32G Ceph Storage
Attacker's Host (H1, H2, H3)	Hping3, T50	VM; Ubuntu 20.04-64bit, 4GiB RAM, 4Core, 32G Ceph Storage
Legitimate's Host	Iperf, ping	VM; Ubuntu 20.04-64bit, 4GiB RAM, 4Core, 32G Ceph Storage

Figure 2 is a visual representation of the virtualization environment have built, and it can be shown that OpenvSwitch running on each node has

activated sFlow-agent to monitor network devices. Packet flow sampling and counter sampling are performed by sFlow instances associated with individual Data Sources within the sFlow Agent. The sFlow agent collects counters and packet flow records and sends them in the form of a sflow datagram to sFlow-Collector.

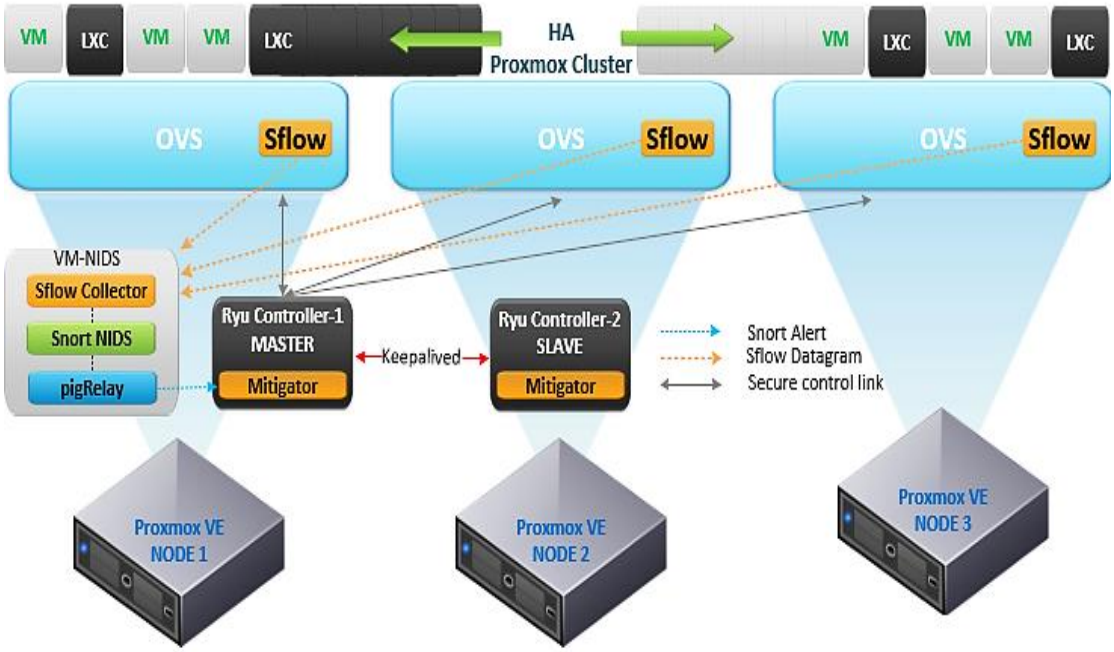

Figure 2. Experiment Infrastructure

VM NIDS contains three modules, namely: sFlow-Collector, Snort-NIDS, and pigrelay. The sFow-Collector acts as a collector of all sent sFlow data from all sFlow-agents; It also converts the sFlow datagram into packet capture (PCAP). In this work, the sampling rate used is 1: 100 and the polling used is 1 second. The 1: 100 sampling rate means that one packet sample is taken from every 100 packets captured by the sFlow-agent, with 1-second intervals on the sampling counters, and only the header information of the corresponding sample is sent to the collector.

Snort NIDS acts as an attack detection and alarm mechanism and operates according to the rules have created to trigger the alarm in a DDoS attack. Snort NIDS is not run inline on the network interface to analyze traffic on the captured network but instead runs in packet reading mode. This process runs parallel with converting process sFow data to PCAP, as shown in the system workflow for every received packet in Figure 4. Pigrelay Module acts as a relay that listens for alert messages from snort via the Network Socket and sends alert messages to the Controller. Apart from implementing the High Availability feature in the Proxmox virtualization environment, these two controllers also running in dual master-slave mode with Keepalived, with one Controller acting as master and the other as hot standby controller. It is

applied to controllers to achieve Failover and high availability controllers, avoid single points of failure, and ensure continuous operation without any downtime. When Snort NIDS detects a DDoS, an alert message will be sent to the Ryu-Controller on MASTER state, because by default, the MASTER node will be the active node. As shown in Figure 3, the Controller on MASTER state will periodically send a broadcast packet in the form of an Announcement message as information on its availability, allowing all OpenFlow's messages for Switch and Controller communication or vice versa, acknowledged and replied to by the Controller in MASTER state. If the Controller on BACKUP state does not receive the Announcement message more than three times, it indicates that the MASTER Controller is down and initiates a failover process to take over the role of MASTER by sending random ARP packets to the network. All hosts receiving the gratuitous ARP update their tables, which effectively means that a new device owns the virtual IP address on the network.

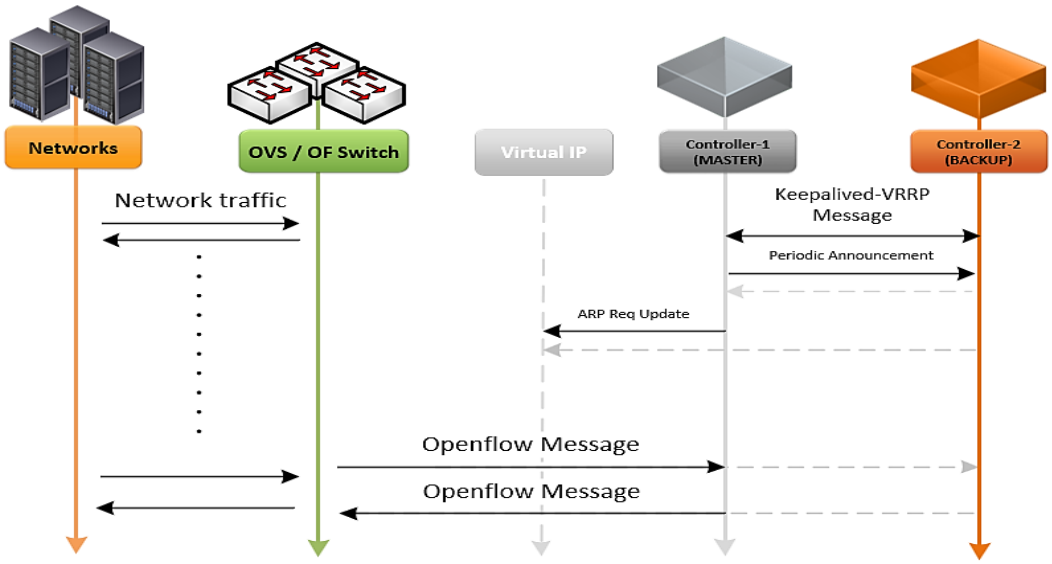

Figure 3. High Availability Ryu Controller using Keepalived

The failback process also works the same way; BACKUP waits for two announcements from the MASTER before giving the virtual IP back and stops sending announcements. All OVS connected to the Controller are configured using the controller's virtual IP, where the setting in the *keepalived.conf* file.

Figure 4 is the system workflow for each packet received to OpenvSwitch. All incoming packets will be carried out in two parallel processes, namely the process of parsing the packet header and then matching it with the flow table and the process of sampling packets into sFlow datagrams. If no match occurs on the flow table, then OpenvSwitch will ask the Controller for a new rule for the new flow. Then, the controller working with the reactive mechanism will respond by sending a new forwarding rule for the new stream. If no match is found in the flow table, this means that

OpenvSwitch already has a flow rule for the received packet. Meanwhile, in another process, the sFlow datagram received by the Collector will be converted back into packet capture data for analysis by Snort NIDS. By its turns, NIDS will only send an alert message to Mitigator if the packet is detected as a DDoS attack packet and send the blocking flow rule to OpenvSwitch.

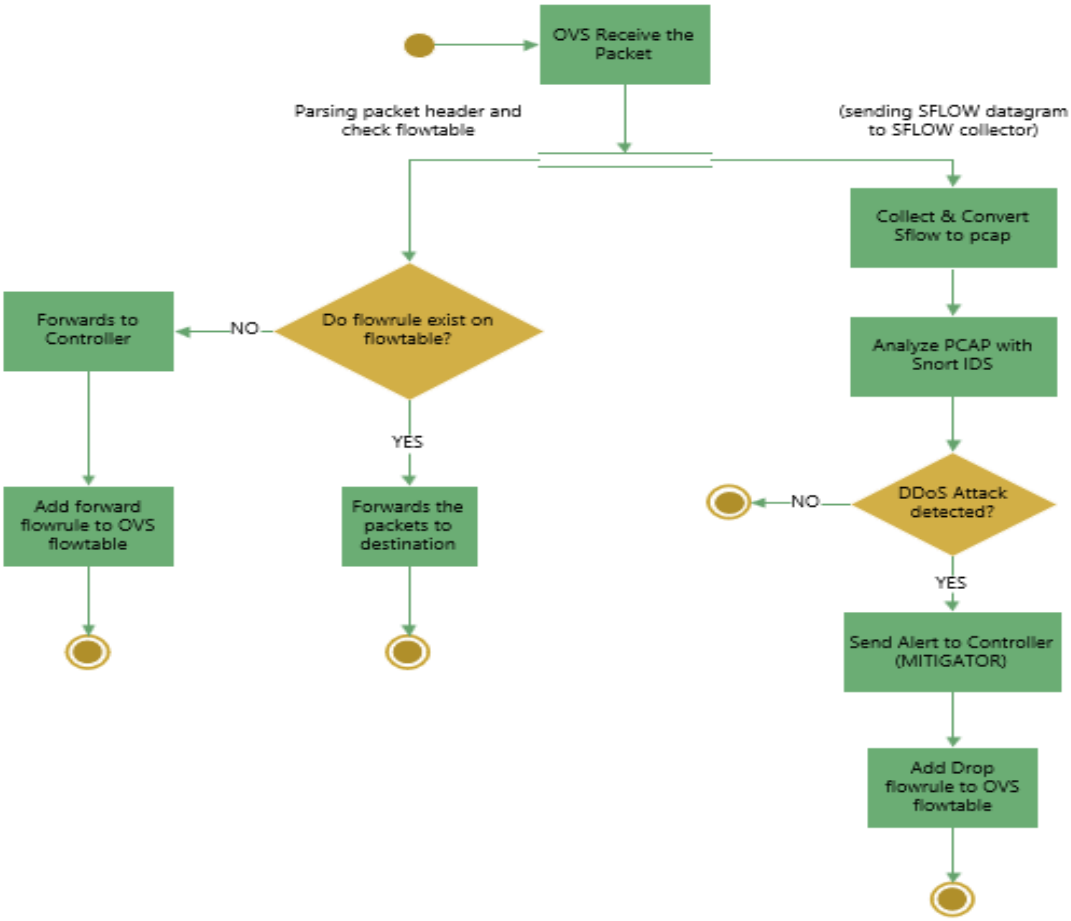

Figure 4. System Workflow

5. EXPERIMENT AND ANALYSIS

In this section, will describe evaluation test and discuss the results obtained. To simulate a DDoS attack, stress test tools used is Hping3 and packet injectors T50 to send TCP / IP packets tailored to the DDoS attack. The types of DDoS attacks that simulate are the top 2 types of DDoS attacks based on the Proportion of DDoS Attack Types in 2019 [1].

5.1 Network Scenarios

The test scenario is divided into three scenarios, wherein in these scenarios, several parameters will be observed; Average Latency, Receive Bitrate, Packet Loss and Mitigation Time.

Table 3. Tests Scenario Summary

Test Scenario	Type	Attack Target	Parameters Monitored
Normal	None	Target Server on Node 1, Node 2, and Node 3	Latency, Bitrate (RX), Packet Loss
Attack Scenario I	UDP Flood	Target Server on Node 2	Bitrate (RX), Latency, Packet Loss, Mitigation Time
Attack Scenario II	TCP (SYN) Flood	Target Server on Node 3	Bitrate (RX), Latency, Packet Loss, Mitigation Time
HA Controller	Failover, and Failback Test	Controller-1, Controller-2	Latency, Bitrate (RX), Packet Loss

Figure 5(a) and 5(b) shows a test attack scenario I attack scenario II on the testing infrastructure, where three hosts are prepared as the attacker's host and three hosts as target servers running on a virtual machine at each node.

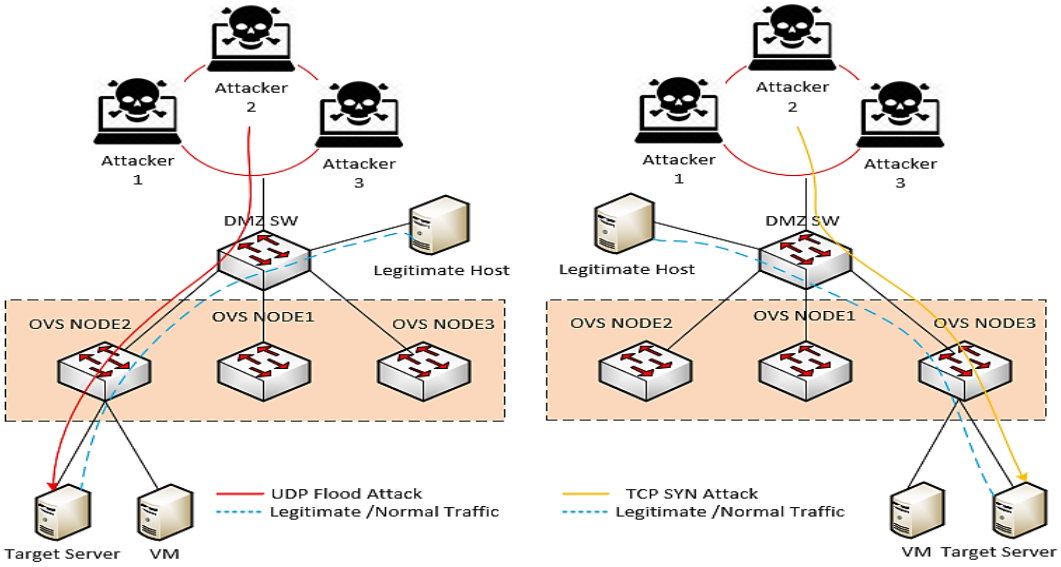

Figure 5. DDoS Attack scenario on virtualization environment

In implementing the DDoS attack mitigation system, using mitigator module, as shown in algorithm 1. The Controller will read the alert message sent from Snort NIDS, then perform parsing and data extraction of the message packet and send the drop flow to the OpenFlow switch.

Algorithm 1 : Mitigator Module

```
1  Mitigation begin
2      AM ← Alert Message from Snort
3      Line ← Pointer to the first AM
4      PM ← list of all Packet Message
       /* EoF is the end of AM
5      while Line ≠ EoF do
6          if PM contains any source IP, source MAC, destination IP and IP protocol from AM then
7              Extract the corresponding srcIP, srcMAC, dstIP, ipProto from PM
8              Add drop flow : srcIP, srcMAC, destIP and ipProto
9          end
10     end
11 end
```

Algorithm 1. Mitigator module

5.2 Test Result

As previously explained, in this scenario, the system is monitored normally on all virtualized environment nodes. Figure 6 is a graphical representation of the trend of network traffic on one of the OVS.

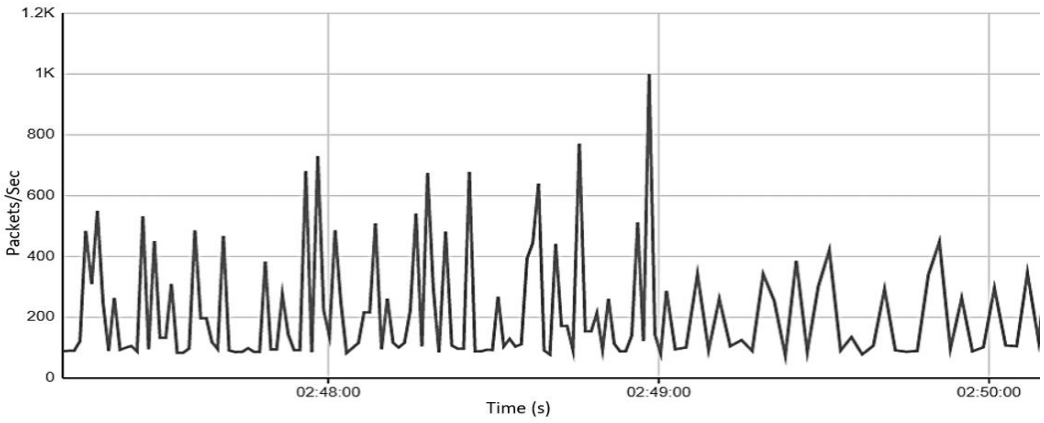

Figure 6. Normal traffic on each Node before DDoS Attack

Table 4 shows some value of *Quality of Service* (QoS) parameters, from hosts 1, 2, and 3 (attacker's hosts) and 4th host (legitimate host) to each target server located at each node. Normally all hosts have an average latency value of <1ms and an average bandwidth of >800 Mbit/s.

Table 4. System Performance on each Node

Host No	Server on Node 1			Server on Node 2			Server on Node 3		
	Average Latency	Bitrate (RX)	Packet Loss	Average Latency	Bitrate (RX)	Packet Loss	Average Latency	Bitrate (RX)	Packet Loss
	(ms)	(Mbps)	(%)	(ms)	(Mbps)	(%)	(ms)	(Mbps)	(%)
1	0,864	884	0	0,821	886	0	0,868	885	0
2	0,826	871	0	0,923	915	0	0,874	891	0
3	0,858	882	0	0,916	909	0	0,901	903	0
4	0,914	909	0	0,934	881	0	0,896	898	0

To find out the impact of a DDoS attack, we also capture and plot the result (Figure 7) of DDoS attack traffic volume when DDoS Defense system is disabled so that network traffic can be shown spikes up to 250,000 packets/s. At the same time, it can also be seen in the image visualization that the CPU utility has increased drastically by 25% on one of the nodes that were getting a DDoS attack. This shows that a DDoS attack in the virtualization environment will impact consuming bandwidth and available resources.

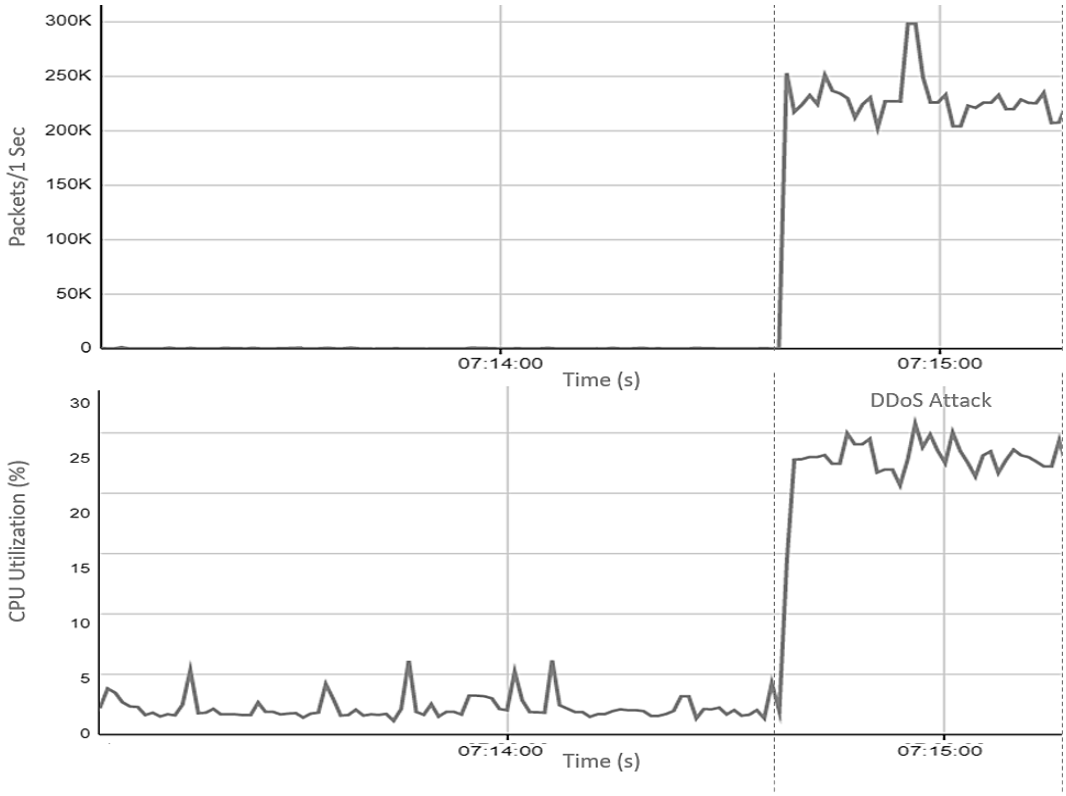

Figure 7. Target's server under attack - without DDoS Defense

5.2.1 Attack Scenario I

The attack scenario to be executed in this section is the UDP Flood attack, where this attack is carried out by carrying out a DDoS attack originating up to 3 attacker's hosts, as shown in Figure 5(a). The stress test tool used in this scenario is hping3 which is executed on the command "hping3 -flood -UDP -k -s 53" by flooding the UDP port 53 packets to the target server (172.16.200.200), which is migrated to Node-2. From Figure 8, it can be observed that the traffic was normal up to 80 seconds, then three attacking hosts started launching UDP flood attacks which were detected since the number of packets peaked drastically up to 50,000 packets/s. DDoS attacks in the form of UDP floods can be mitigated to the 85th seconds, and the network returns to its normal state afterward.

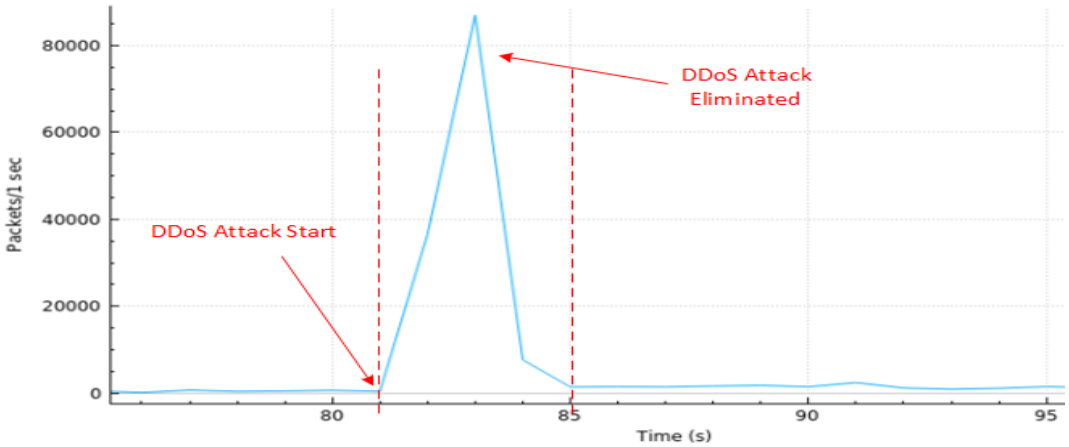

Figure 8. Mitigation Time - UDP flood (3 attackers)

Table 5 shows the average latency and packet loss results from host-4 (legitimate host) to the server that targets a UDP flood attack. It can be shown that the mitigation time increases as the number of attack sources increases.

Table 5. Performance Result – UDP flood mitigation

No. of Host (Attacker)	Without DDoS Defense			With DDoS Defense			
	Average Latency	Packet Loss	Bitrate (RX)	Average Latency	Packet Loss	Bitrate (RX)	Mitigation Time
	(ms)	(%)	Mbit/s	(ms)	(%)	Mbit/s	(s)
1	0,88	2,2	89	0,732	0	876	2
2	0,90	2,7	56	0,76	0	881	3
3	0,93	3,3	32,5	0,771	0	874	4

5.2.2 Attack Scenario II

The second attack scenario is the TCP SYN attack, launched by 1 to 3 attackers' hosts, as shown in Figure 5(b). The packet injector used in this scenario is T50 which is executed with the command "t50 --flood --turbo --dport 80 -S --protocol TCP" send TCP SYN request connection faster to port 80 on the target server migrating to Node-3.

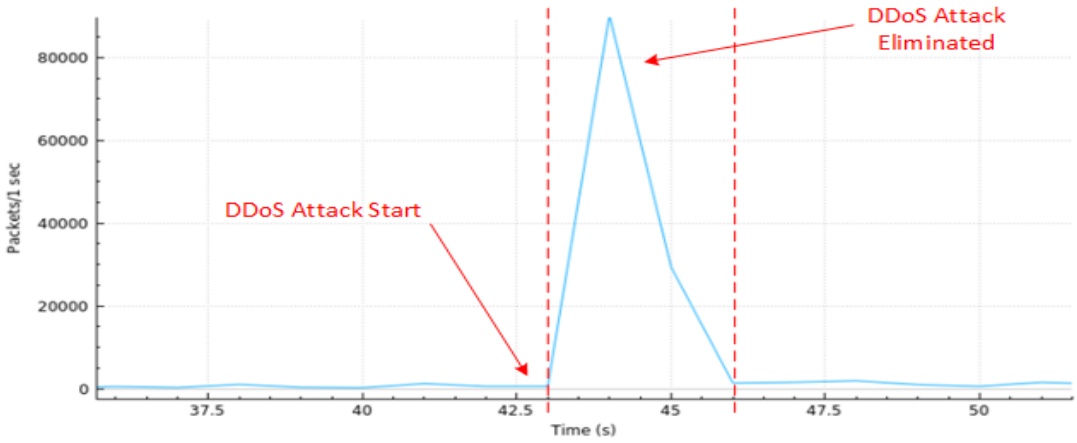

Figure 9. Mitigation Time – TCP SYN flood (1 attacker)

From Figure 9, it can be observed that a DDoS attack with one attacking host was detected at 43 seconds, the overflow of TCP SYN packets increased up to 80,000 packets/s, and the network returned to normal at 46 seconds. Table 6 shows the DDoS Defense system's performance, can be observe that the average latency and mitigation time of the TCP SYN flood attack increases based on the number of attack sources and does not find any packet loss values.

Table 6. Performance Result – TCP (SYN) flood mitigation

No. of Host (Attacker)	Without DDoS Defense			With DDoS Defense			
	Average Latency	Packet Loss	Bitrate (RX)	Average Latency	Packet Loss	Bitrate (RX)	Mitigation Time
	(ms)	(%)	Mbit/s	(ms)	(%)	Mbit/s	(s)
1	1,4	19	4,4	0,819	0	858	3
2	1,99	20,6	2,0	0,883	0	862	3
3	2,37	23,3	0,3	1,075	0	856	4

5.2.3 HA Controller Test

In this scenario, High Availability test on the Controller by measuring when a Failover and Failback process occurs between Controller-1 and Controller-2. As shown in Figure 10, Controller-1 is the Primary Controller, and the setting has a higher priority than Controller-2 as the Secondary Controller. Based on the observation, the transition process of MASTER and BACKUP conditions during Failover takes up to 2 seconds, and Failback is 2 seconds.

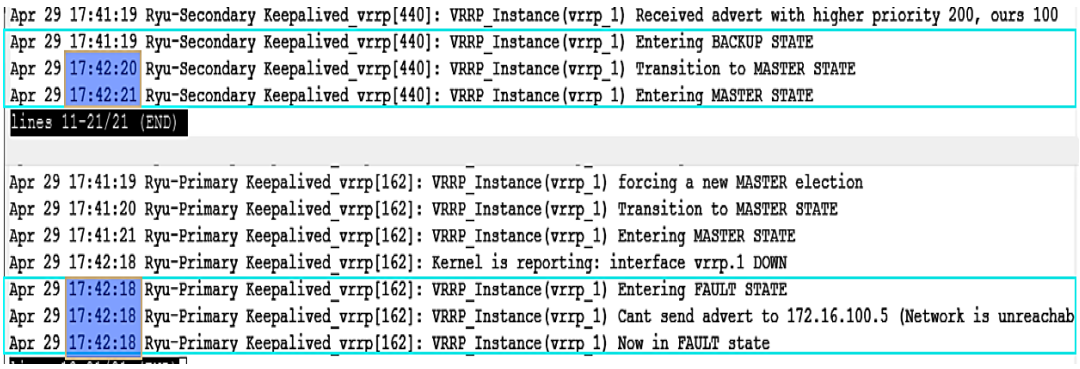

Figure 10. Failover and Failback Controller

Based on the results of Quality of Service (QoS) measurements during the Failover and Failback processes, table 7 shows that no Packet Loss was recorded in the communication of the data plane.

Table 7. Performance results – Failover and Failback Test

Test No.	Failover			Failback		
	Average Latency (ms)	**Packet Loss** (%)	**Bitrate (RX)** (Mbit/s)	**Average Latency** (ms)	**Packet Loss** (%)	**Bitrate (RX)** (Mbit/s)
1	0,739	0,00	885	0,681	0,00	884
2	0,678	0,00	891	0,738	0,00	903
3	0,704	0,00	903	0,638	0,00	882
Average	0,707	0,00	893	0,686	0,00	890

The comparison results with previous studies can be seen in table 8. In the UDP Flood scenario, the average mitigation time is 3.0 seconds; This result is very close to the same test conducted by Manso et al. [3], the average mitigation time obtained is 3.07 seconds. While in the second scenario with the TCP SYN Flood attack, the average mitigation time is 3.3 seconds. This shows a better mitigation time than the results obtained by Sumantra et al. [13], which is 10.2 seconds.

Table 8. Comparison to Existing Research

Attack Type	**This research study**	**Existing Research** (Manso et al. [4])	**Existing Research** (Sumantra et al. [14])
	Average Mitigation Time	Average Mitigation Time	Average Mitigation Time
	(s)	(s)	(s)
UDP Flood	3,0	3,07	N/A
SYN Attack	3,3	N/A	10,2

And besides that, it also shows that by using sampled flow (sFlow), the mitigation time achieved is almost the same as IDS integration using Network TAP; even the mitigation time is even faster when compared to the Anomaly-based technique using a statistical approach.

6. CONCLUSION

This paper presents an SDN-based DDoS attack defense system in a virtualization environment and evaluates it in a production testbed of the Proxmox-VE. In this research study, the approach used considers the flexibility and scalability of the virtualization environment by adopting sampling techniques integrated with network intrusion detection systems (NIDS). The results show that DDoS defense system can effectively detect and mitigate UDP flood and TCP SYN Flood attacks in a Virtualization Environment. Besides, by implementing sFlow, which is integrated with Snort-IDS, this architecture can also be extended and improved to identify other DDoS attacks. In future work will cover to use sFlow for hybrid technique that combines signature-based and anomaly-based techniques using machine learning to improve DDoS attack detection capabilities in the Cloud Computing Environment.

Acknowledgments

The author would like to thank the Ministry of Education and Culture, the Directorate General of Vocational Education, and Diksi Resources for their support through the 2019 PTNB Affirmation Scholarship Program.

REFERENCES

[1] A. P. Utomo, I. Winarno, and I. Syarif, "**Towards a Resilient Server with an external VMI in the Virtualization Environment**," *Emit. Int. J. Eng. Technol.*, vol. 8, no. 1, pp. 49–66, Jun. 2020, doi: 10.24003/emitter.v8i1.468.

[2] Q. Yan and F. R. Yu, "**Distributed denial of service attacks in software-defined networking with cloud computing**," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 52–59, Apr. 2015, doi: 10.1109/MCOM.2015.7081075.

[3] M. Hao, "**2020 Mid-Year DDoS Attack Landscape Report-3**," NSFOCUSGLOBAL, 3, Aug. 2020.

[4] P. Manso, J. Moura, and C. Serrão, "**SDN-Based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks**," *Information*, vol. 10, no. 3, p. 106, Mar. 2019, doi: 10.3390/info10030106.

[5] S. Badotra and S. N. Panda, "**SNORT based early DDoS detection system using Opendaylight and open networking operating system in software defined networking**," *Clust. Comput.*, May 2020, doi: 10.1007/s10586-020-03133-y.

[6] Maxli Campos and J. S. B. Martins, "**A Sdn-Based Flexible System For On-The-Fly Monitoring And Treatment Of Security Events**," Jan. 2017, doi: 10.5281/ZENODO.1291094.

[7] Po-Wen Chi∗, Chien-Ting Kuo∗†, and He-Ming Ruan∗, "**An AMI Threat Detection Mechanism Based on SDN Networks**," *Secur. 2014 Eighth Int. Conf. Emerg. Secur. Inf. Syst. Technol.*, no. 8, p. 208, 2014.

[8] A. Yazdinejadna, R. M. Parizi, A. Dehghantanha, and M. S. Khan, "**A kangaroo-based intrusion detection system on software-defined networks**," *Comput. Netw.*, vol. 184, p. 107688, Jan. 2021, doi: 10.1016/j.comnet.2020.107688.

[9] M. A. Lopez, D. M. Ferrazani Mattos, and O. C. M. B. Duarte, "**An elastic intrusion detection system for software networks**," *Ann. Telecommun.*, vol. 71, no. 11–12, pp. 595–605, Dec. 2016, doi: 10.1007/s12243-016-0506-y.

[10] P. M. Ombase, S. T. Bagade, N. P. Kulkarni, and A. V. Mhaisgawali, "**DoS Attack Mitigation Using Rule Based and Anomaly Based Techniques in Software Defined Networking**," p. 7, 2017.

[11] S. Wang *et al.*, "**SECOD: SDN sEcure control and data plane algorithm for detecting and defending against DoS attacks**," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, Taipei, Apr. 2018, pp. 1–5. doi: 10.1109/NOMS.2018.8406196.

[12] N. I. G. Dharma, M. F. Muthohar, J. D. A. Prayuda, K. Priagung, and D. Choi, "**Time-based DDoS detection and mitigation for SDN controller**," in

2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS), Busan, South Korea, Aug. 2015, pp. 550–553. doi: 10.1109/APNOMS.2015.7275389.

[13] M. Latah and L. Toker, "**A novel intelligent approach for detecting DoS flooding attacks in software-defined networks**," *Int. J. Adv. Intell. Inform.*, vol. 4, no. 1, p. 11, Mar. 2018, doi: 10.26555/ijain.v4i1.138.

[14] I. Sumantra and S. Indira Gandhi, "**DDoS attack Detection and Mitigation in Software Defined Networks**," in *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*, Pondicherry, India, Jul. 2020, pp. 1–5. doi: 10.1109/ICSCAN49426.2020.9262408.

[15] S. Usman, I. Winarno, and A. Sudarsono, "**Implementation of SDN-based IDS to protect Virtualization Server against HTTP DoS attacks**," in *2020 International Electronics Symposium (IES)*, 2020, pp. 195–198.

[16] A. Leal, J. F. Botero, and E. Jacob, "**Improving Early Attack Detection in Networks with sFlow and SDN**," in *Applied Computer Sciences in Engineering*, vol. 916, J. C. Figueroa-García, J. G. Villegas, J. R. Orozco-Arroyave, and P. A. Maya Duque, Eds. Cham: Springer International Publishing, 2018, pp. 323–335. doi: 10.1007/978-3-030-00353-1_29.