Iraqi Journal of Information and Communications Technology(IJICT)
Vol. 4, Issue 1, March 2021
ISSN:2222-758X

DDOS ATTACK DETECTION AND MITIGATION AT SDN ENVIROMENT

Huda S. Abdulkarem [1], **Ammar D. Alethawy** [2]
[1,2] College of Information Engineering, Al-Nahrain University, Baghdad, Iraq
{huda.saleh, ammar.alaythawy}@coie-nahrain.edu.iq [1,2]
Received: 07/08/2020 , Accepted:18/09/2020

***Abstract*- Software- Defined Networking (SDN) is a network architecture approach enables the network to be intelligently and centrally controlled, or programmed, using software applications to helps operators manage the entire network consistently and holistically. Although SDN on the surface provides a simple framework for network programmability and monitoring, few have been said about security measures to make it resilient to hitherto security flaws in the traditional networks and the new threats the architecture is ushering in. One of the security weaknesses the architecture is ushering in due to separation of control and data plane is distributed Denial of Service (DDoS) attack. The main goal of this attack is to make network resources unavailable to legitimate users or introduce large delays. In this paper, the effect of a DDoS attack on SDN is presented, and automatic detection and mitigation of the attacks, using an SDN application written in python with the help of the OpenFlow protocol.**

keywords: **SDN, DDoS, OpenDayLight, OpenFlow.**

I. INTRODUCTION

Distributed denial- of- service (DDoS) is the most popular attack making network devices cannot be accessed or used by legitimate users [1]. DDoS attacks make online services unavailable by flooding victims with traffic from multiple attackers [2]. One of the most prominent features of SDN is the division of the data plane and the control plane as shown in Fig. 1. The control plane makes the decisions where to send packets, and the data plane implements these decisions and actually forwards the packets [3]. Additionally, optimal network operations can be achieved by employ the benefits of centralization i. e. making decisions with a complete view of overall network conditions from a single and centralized point [4]. The application layer resides on top of the control plane and is implemented by means of REST APIs. The control plane communicates with the data plane byways as defined by an SDN standard known as OpenFlow. OpenFlow is, therefore, a communication protocol between an SDN controller and physical switches. From an application standpoint, OpenVSwitch (OVS) is an open- source switching software that uses OpenFlow and is supported also by cloud platforms such as OpenStack, Mininet as virtual network emulation software [5]. The work proposes a feasible SDN-based generic solution to mitigate DDoS attacks without adding too much overload on the network. The current proposal aims to detect and mitigate the DDoS attack when an online server is under attack. Briefly, the proposed solution has the following characteristics:

1) It compares at runtime the expected trend of normal traffic against the trend of monitored traffic;
2) If a significant deviation on the traffic trend is detected, then an event is created;
3) As an event associated with a DDoS attack is produced, then an SDN application programmed to start capturing and analyzing the traffic to creates flow rules for blocking the malign traffic with OpenDayLight controller.

ISSN:2222-758X

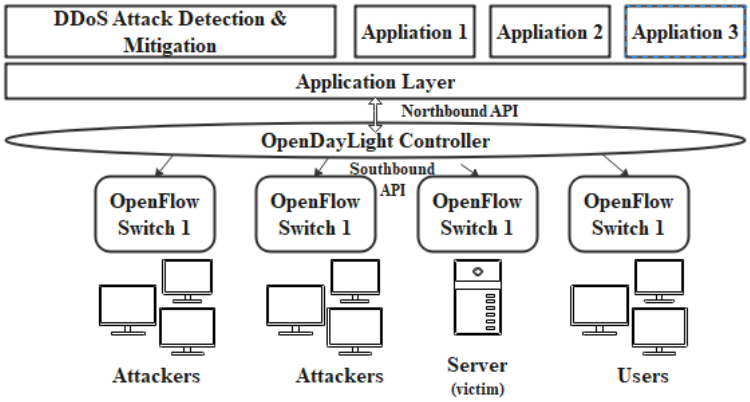

Figure 1: Software Defined Networking (SDN) architecture with DDoS attack Application

II. Literature Survey

In paper [6], the effect of the DoS attack on SDN is presented using Mininet, OpenDaylight (ODL) controller. (ICMP) flood attack is performed on a (TCP) server and a (UDP) server which are both connected to OpenFlow switches. The simulation results reveal a drop in network throughput from 233 Mbps to 87.4Mbps and the introduction of large jitter between 0.003 ms and 0.789 ms during a DoS attack. This paper offers the impact of the DoS attack on SDNs without the mitigation of the attack. [7] In this paper, proposed SDN-Guard, a comprehensive SDN solution that is able to mitigate SDN- specific threats related to DoS attacks. By dynamically rerouting potential malicious traffic, adjusting flow timeouts, and aggregating flow rules associated with malicious traffic. In these two papers only study DOS, not DDoS attack first one only impact and the second take ten minutes to mitigation. [2] In this paper, studied how to utilize SDN to detect DDoS attacks. The methods capture the flow volume feature as well as the flow rate asymmetry feature, to adaptively change the flow monitoring granularities on all switches to quickly locate the potential victims and suspicious attackers. [8] In this paper, propose to use Advanced Support Vector Machine (ASVM) algorithm in order to detect DDoS attacks. Both papers in [2] and [8] talk only about DDoS detection by using POX and ONOS controllers. In [9], a firewall will block the addresses forwarded by the server. DDoS detection method checks the incoming traffic and analyzes it. If an attacker is found, the address will be forwarded to the firewall. The firewall will mount and block the packets. This paper uses a third- party firewall for blocking, with four hosts and one server, using the POX controller. This solution has no different from traditional network solutions which forwarding traffic to centralize firewall. [10] In this paper, show how a DDoS attack can be instigated on a primary server. The attack is instigated by generating a huge number of packets with destination IP addresses to switch which connects the primary server. The flow table rules are repeatedly installed by the (POX) controller into this switch, leading to exhaustion of its flow table space. The paper contains a weak attack scenario, the flow entries for the packets which are directly installed by the controller are classified as attack traffic whereas the flow table entries for those packets which are requested by the switch are grouped under genuine traffic, by manipulating with "TIMEOUT" parameter value without SDN application. [11] In this paper, the network constructs consist of virtual

hosts, representing both normal users and attackers. The goal is to enable SDN programmability, against a DDoS ping attack. Shows TCP and UDP throughputs, round- trip time as measurable by emulated network users to demonstrate the application of SDN in resolving the attacking adverse effects with Opendaylight controller. Paper in [11] is the most similar to this project idea. with different in testbed components, its use GNS3 emulator and VMware where this paper use Mininet support OpenFlow for highly flexible custom routing and Software- Defined Networking. The network topology of [11] does not customize like this paper topology its only simple mesh topology. The weakness in paper [11], it already knows the attacker IP while this paper finds and extracts the attackers from the traffic toward the victim. In [11] the dropping rule send to all the switches in the network so the user will be denied from all the other network, while this project perform rule sending to specific switch, in [11] the script work manually so there is no time for mitigation can be evaluated . In this paper python script work automatically whenever attack detected

III. OpenFlow Protocol

OpenFlow is a Layer 2 communications protocol that gives access to the forwarding plane of a network switch or router over the network. The OpenFlow pipeline of every OpenFlow switch contains multiple flow tables, each flow table containing multiple flow entries [12]. A flow table consists of flow entries, Match Fields, Priority, Counters, Instructions, Timeouts and Cookie.An OpenFlow switch is an OpenFlow- enabled data switch that communicates over the OpenFlow channel to an external controller [13]. It performs packet lookup and forwarding according to one or more flow tables and a group table as shown in Fig. 2. The OpenFlow switch communicates with the controller and the controller manages the switch via the OpenFlow switch protocol. They are either based on the OpenFlow protocol or compatible with it. There is a set of actions that the controller will send to the switch to perform; forward, drop, push in a queue, quality of service and modifying a field, i. e. , modifying VLAN tag, MAC address or IP address [14].

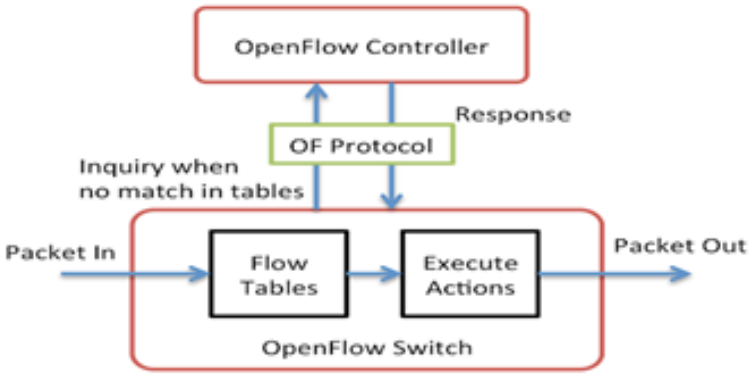

Figure 2: OpenFlow protocol

IV. Implementation

This section discusses the designing and implementation of the proposal

ISSN:2222-758X

A. Architecture overview of the tested Components

The system operates via a loop control among three basic architectural components in Fig. 3 part- A contains the Open-DayLight controller, part- B include the network topology, and part-C contain SDN application. The testbed components, namely, mininet emulator and OpenDayLight, installed with the Ubuntu operating system as shown in Fig. 4. VirtualBox is in this testbed used with two images. The first virtual image runs the Ubuntu-based Linux operating system with the SDN controller, namely, (ODL) Opendaylight with an IP address of 192.168.56.6. A Python script has been developed to send through the northbound REST API of ODL a necessary commands to preplan all the OVS forwarding tables as well as to drop the attack traffics, once detected. The second virtual image of VMware is used as a mininet VM (IP 192.168.56.3) to create a network of virtual hosts, switches, and links. Mininet hosts run standard Linux network software, and its switches support OpenFlow for highly flexible custom routing and Software- Defined Networking.

- Network topology consists of sixty-four hosts, named as Host1 to Host64 (with the network address from 10.0.1.0/24 to 10.0.8.0/24 each network related to one switch).
- Attackers are some of these hosts who want to disrupt the connectivity between users and the victim server.
- The victim is a server connected to switch number two with the IP address (10.0.2.60).

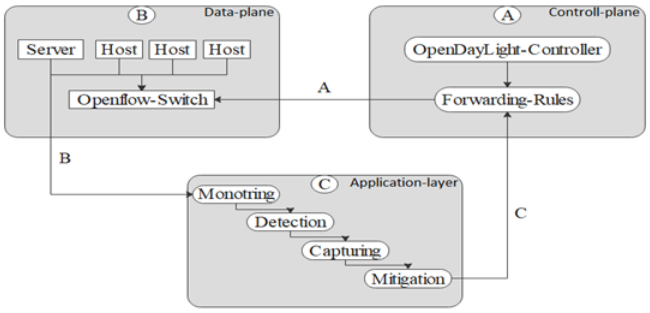

Figure 3: Architecture overview of the testbed components

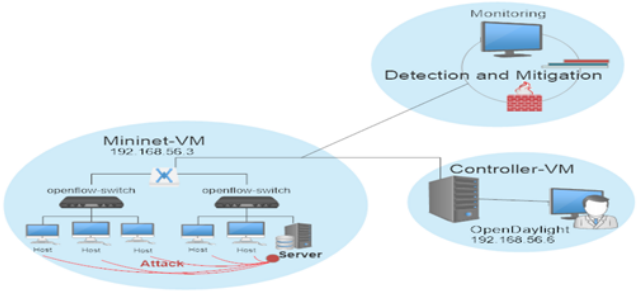

Figure 4: The system's conceptual model

ISSN:2222-758X

B. Experimental set- up

As shown in Fig. 5, the mininet emulator shows the emulated topology consisting of eight OVS nodes connect to the central ODL controller. By creating custom Mininet topologies by using Python API. The

```
$ sudo mn --custom ~/mininet/custom/topo.py
```

```
from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch, UserSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel
from mininet.link import Link, TCLink
 def topology():
            net = Mininet( controller=RemoteController, link=TCLink, switch=OVSKernelSwitch )
          # Add hosts and switches
         h1 = net.addHost( 'h1', ip="10.0.1.10/24", mac="00:00:00:00:00:01" )
         h2 = net.addHost( 'h2', ip="10.0.1.20/24", mac="00:00:00:00:00:02" )
```

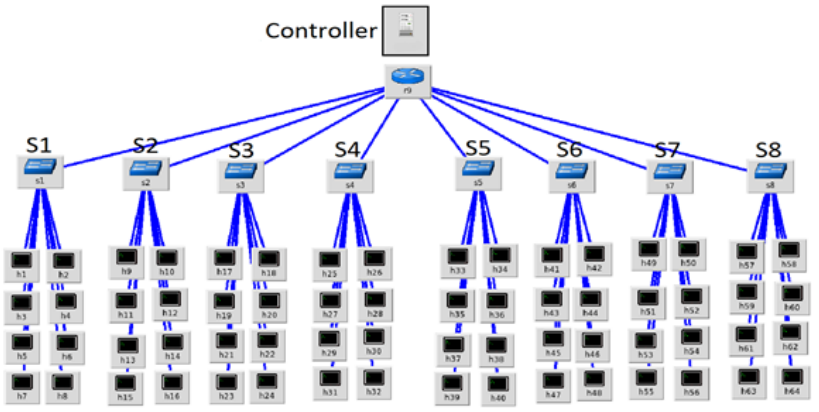

Figure 5: Network topology in mininet

C. Scenario of attack

The scenario assumes that DDoS attack should be preferably mitigated, to avoid the server suffering from the negative performance effects induced by a DDoS attack. The evaluation if the main ideas behind the proposal are completely fulfilled. Therefore, it is not the intention to perform test scenarios with thousands of nodes, a network simulator would be a more convenient tool. Nevertheless, this option is out of the scope of the current work. With the intent of validating the most relevant functional aspects of the proposal. Host 1, host 42, and host 64 are randomly selected for attacking the server, and host 12 measures the reachability to the server and service performance before and after the botnets attack. Fig. 6 shows the server traffic when there is no attack.

ISSN:2222-758X

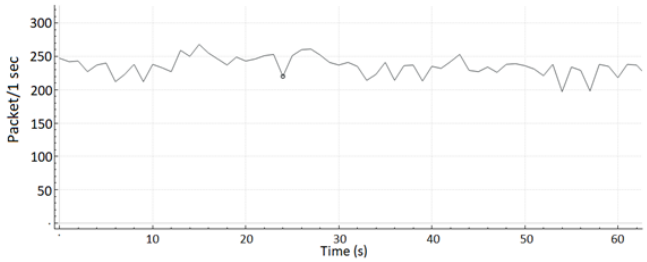

Figure 6: Server rate when there is no attack

Hosts will attack the server using the Hping3 tool. Hping3 is a network tool able to send custom TCP/ IP packets. It supports TCP, UDP, ICMP, and RAW- IP protocols, Hping3 allowing to send manipulated packets. This tool allows controlling the size, quantity, and fragmentation of packets to overload the target and bypass or attack firewalls.

♯hping3 -V -1 -d 1400 - -faster 10.0.2.60

Volumetric attack (bandwidth attack) will be used (ICMP flood) against the victim, Volumetric DDoS attacks are designed to overwhelm internal network capacity with significantly high volumes of malicious traffic. These attacks attempt to consume the bandwidth of the target server (10.0.2.60). To give a better idea of the QoS impact in hosts while the network was under attack. As shown in Fig. 7, the results of the average ICMP response time were retrieved from the ping command of host12 toward the server.

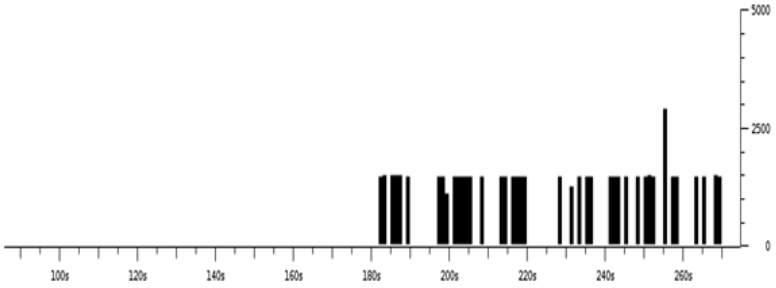

Figure 7: Host 12 ICMP response time during the attack

D. Proposed Solution

Current practices in traditional networks would be to rely on a firewall sitting at the network domain border or gateway to drop harmful packets. With SDN All the OVS switches can be reprogrammed to drop attacker traffics at the earliest possible locations, by using an SDN application (python script was written and developed in this paper) to capture and analyze the traffic toward the server (victim). whenever a detection of unusual behavior happens in the traffic, application start packet analyzing to extract the attackers IP according to the largest traffic senders, who forwarding huge traffic, for making normal users easily reach the server, a dropping rule applies to the Openvswitch the server was connected to as shown in Fig. 8 .The rule was forwarded from the application to OpenDayLight controller API. When Fig. 9 show the

server traffic before and after the attack. The ovs- ofctl command- line tool used to monitoring and administering OpenFlow in the switch. It can also show the current state of an OpenFlow switch, including features, configuration, and table entries. The first three flow entries rules with high priority were added from the application to block malicious hosts. Applying another scenario with different number of host or switch, the perform of the application didn't change, because it depend on the victim location in the network to apply the blocking rule on it.

```
root@mininet-vm:~# sudo ovs-ofctl --o dump-flows s2
NXST_FLOW reply (xid=0x4):
 cookie=0x0,duration=2340.97s,table=0,n_packets=0,n_bytes=0,idle_age=2340,
priority=1000,ip,nw_src=10.0.6.0/24 actions=drop
cookie=0x0,duration=2340.97s,table=0,n_packets=146971,n_bytes=211922774,idle_age=43,p
riority=1000,ip,nw_src=10.0.1.0/24 actions=drop
 cookie=0x0, duration=2340.947s, table=0, n_packets=0, n_bytes=0, idle_age=2340,
priority=1000,ip,nw_src=10.0.8.0/24 actions=drop
 cookie=0x0, duration=1032.778s, table=0, n_packets=0, n_bytes=0, idle_age=1032,
priority=10,ip,nw_dst=10.0.2.30 actions=output:4
 cookie=0x0, duration=1032.767s, table=0, n_packets=0, n_bytes=0, idle_age=1032,
priority=10,ip,nw_dst=10.0.2.60 actions=output:7
idle_age=1032, priority=10,ip,nw_dst=10.0.2.10 actions=output:2
 idle_age=1032, priority=65535,ip,dl_dst=00:00:00:00:01:02 actions=output:1
 cookie=0x0, duration=1032.793s, table=0, n_packets=10, n_bytes=420, idle_age=61,
priority=1,arp actions=FLOOD
```

Figure 8: Openflow table entries at Openvswich

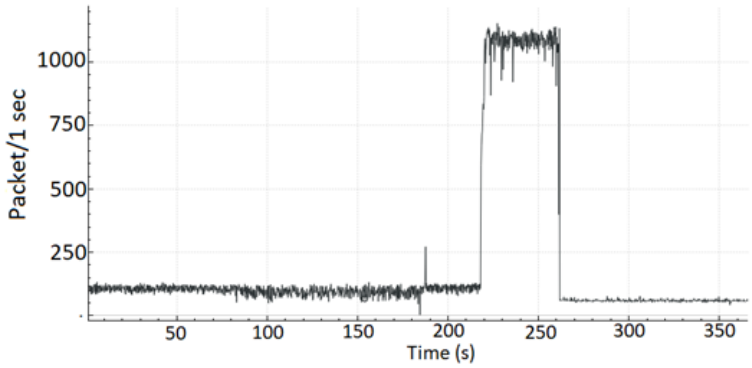

Figure 9: Server rate after mitigation attack

V. CONCLUSIONS

This paper propose detects DDoS attack and mitigates the negative consequences of the widespread effect of that attack on potential victims. The conducted attacks used ICMP flooding in a simple scenario. The system was able to detect and mitigate DDoS with an average time of 100- 150 seconds. There are different methods for detecting and mitigating attacks, each method has different used this paper focusing on a solution that works particularly well for the SDN environment, based on its specifications, points of strength, and limitations by using the fact that the openvswitch could be programmed. The application giving the ability to add new features and ideas in the FUTURE BEHIND the advantage of adding Openflow

ISSN:2222-758X

entries rules to perform an enhanced SDN environment with more functions. In future research, one could investigate relevantly in queues for rate-limiting and for QoS implementation, rather than drop action rule.

REFERENCES

[1] B. zhang, T. Zhang and Z. Yu,"DDoS Detection and Prevention Based on Artificial Intelligence Techniques", 3rd IEEE International Conference on Computer and Communications (ICCC), Chengdu, China, pp. 1276-1280, 13-16 Dec. 2017, DOI. 10.1109/CompComm.2017.8322748.

[2] Y. Xu, Y. Liu,"DDoS Attack Detection under SDN Context", in IEEE INFOCOM, 35th Annual IEEE International Conference on Computer Communications, San Francisco, CA, USA, pp. 1-9, 10-14 April 2016, DOI. 10.1109/INFOCOM.2016.7524500.

[3] W. H. Muragaa, K.Seman and M. F. Marhusin, "A POX Controller Module to Prepare a List of Flow Header Information Extracted from SDN Traffic", World Academy of Science, Engineering and Technology,International Journal of Computer and Systems Engineering, vol. 11, no. 12, pp. 1305-1308, 2017, DOI. 10.5281/zenodo.1314781.

[4] S. Ali, M. Khalid Alvi, S. Faizullahy and Asad Khan, " Detecting DDoS Attack on SDN Due to Vulnerabilities in OpenFlow", International Conference on Advances in the Emerging Computing Technologies (AECT), pp. 1-6, 10 Feb 2020, DOI. 10.1109/AECT47998.2020.9194211.

[5] P. Manso, J. Moura and C. Serrao, "SDN- Based Intrusion Detection System for Early Detection and Mitigation of DDoS Attacks", Information Sciences, Technologies and Architecture Research Center (ISTAR- IUL), ISCTE- Instituto Universitario de Lisboa, 1649- 026 Lisbon, Portugal, pp. 1-17, 8 Mar. 2019, DOI. 10.3390/info10030106.

[6] A. Sangodoyin, T. Sigwele, "DoS Attack Impact Assessment on Software Defined Networks", ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pp. 11- 22, 2018, DOI. 10.1007/978-3-319-76571-62.

[7] L. Dridi, M. Faten Zhani, "SDN- Guard: DoS Attacks Mitigation in SDN Networks", 5th IEEE International Conference on Cloud Networking (Cloudnet), Pisa, Italy, pp. 212-217, 3-5 Oct 2016, DOI. 10.1109/CloudNet.2016.9.

[8] M. Myint Oo, S. Kamolphiwong and T. Kamolphiwong, "The Design of SDN Based Detection for Distributed Denial of Service (DDoS) Attack", 21st International Computer Science and Engineering Conference (ICSEC), Bangkok, Thailand, pp. 258-263, 15- 18 Nov. 2017, DOI. 10.1109/ICSEC.2017.8443939.

[9] R. Mary Thomas, D. James, "DDOS Detection and Denial Using Third Party Application in SDN", International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, pp. 3892-3897, 1-2 Aug. 2017, DOI. 10.1109/ICECDS.2017.8390193.

[10] R. Sanjeetha, A. Prasanna, P. Kumar and A. Kanavali, "Mitigation of Controller induced DDoS Attack on Primary Server in High Traffic Scenarios of Software Defined Networks", IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Indore, India, 16- 19 Dec. 2018, DOI. 10.1109/ANTS.2018.8710066.

[11] S. Sirijaroensombat, C. Phaderm Nangsue and C. Aswakul, "Development of Software- Defined Mesh Network Emulator Testbed for DDoS Defence Study", IEEE 4th International Conference on Computer and Communication Systems (ICCCS),Singapore, Singapore, 23- 25 Feb. 2019.

[12] T. D. Nadeau, K. Gray, "SDN: Software Defined Networks", Oï¿œReilly Media, USA, 2018, pp. 9- 18.

[13] P. G. C. Culver, M. Kaufmann, "Software Defined Networks", 2nd ed, October 2016.

[14] R. Gopakumar, A. M. Unni, "An Adaptive Algorithm for Searching in Flow Tables of OpenFlow Switches", 39th National Systems Conference (NSC), Noida, India, 14-16 Dec. 2015, DOI. 10.1109/NATSYS.2015.7489115.