Cluster Computing (2021) 24:501–513
https://doi.org/10.1007/s10586-020-03133-y

SNORT based early DDoS detection system using Opendaylight and open networking operating system in software defined networking

Sumit Badotra[1] · Surya Narayan Panda[1]

Received: 23 December 2019 / Revised: 11 April 2020 / Accepted: 21 May 2020 / Published online: 29 May 2020

Abstract
Software-defined networking (SDN) is an approach in the network that provides many advantages with the help of separating the intelligence of the network (controller) with the underlying network infrastructure (data plane). But this isolation also gives birth to many security concerns; therefore, the need to protect the network from various attacks is becoming mandatory. Distributed Denial of Service (DDoS) in SDN is one such attack that is becoming a hurdle to its growth. Before the mitigation of DDoS attacks, the primary step is to detect them. In this paper, an early DDoS detection tool is created by using SNORT IDS (Intrusion Detection System). This tool is integrated with popularly used SDN controllers (Opendaylight and Open Networking Operating System). For the experimental setup, five different network scenarios are considered. In each scenario number of hosts, switches and data packets vary. For the creation of different hosts, switches the Mininet emulation tool is used whereas for generating the data packets four different penetration tools such as Hping3, Nping, Xerxes, Tor Hammer, LOIC are used. The generated data packets are ranging from (50,000 per second–2,50,000 per second) and the number of hosts/switches are ranging from (50–250) in every scenario respectively. The data traffic is bombarded towards the controllers and the evaluation of these packets is achieved by making use of Wireshark. The analysis of our DDoS detection system is performed on the basis of various parameters such as time to detect the DDoS attack, Round Trip Time (RTT), percentage of packet loss and type of DDoS attack. It is found that ODL takes minimum time to detect the successful DDoS attack and more time to go down than ONOS. Our tool ensures the timely detection of fast DDoS attacks which delivers the better performance of the SDN controller and not compromising the overall functionality of the entire network.

Keywords DDoS attacks · SDN controllers · Opendaylight · Open network operating system · Security

1 Introduction

SDN is the next-generation automation in the networks and is also known as the paradigm shift in the computer networking industry. SDN has decoupled the control plane and data plane in the network and has centralized the control plane work in the controller [1, 2]. Before SDN, traditional networking revolves around devices having control planes and data planes integrated into a single device which was changed with the SDN [3].

Control Plane is used to create the network paths and then give the instructions to the data plane which is also on the same device and it uses the paths from the control plane and takes packets from source to destination [4]. Traditional network devices like Switch, Routers have control plane and data plane inbuilt in their systems [5]. For routers, routing protocols like Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), Routing Information Protocol (RIP), etc. acts as the control plane and Forwarding Information Base (FIB) or Cisco Express Forwarding (CEF) acts as the data plane [6]. With centralization, lots of benefits arise in the network

✉ Sumit Badotra
summi.badotra@gmail.com

Surya Narayan Panda
snpanda@chitkara.edu.in

[1] Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India

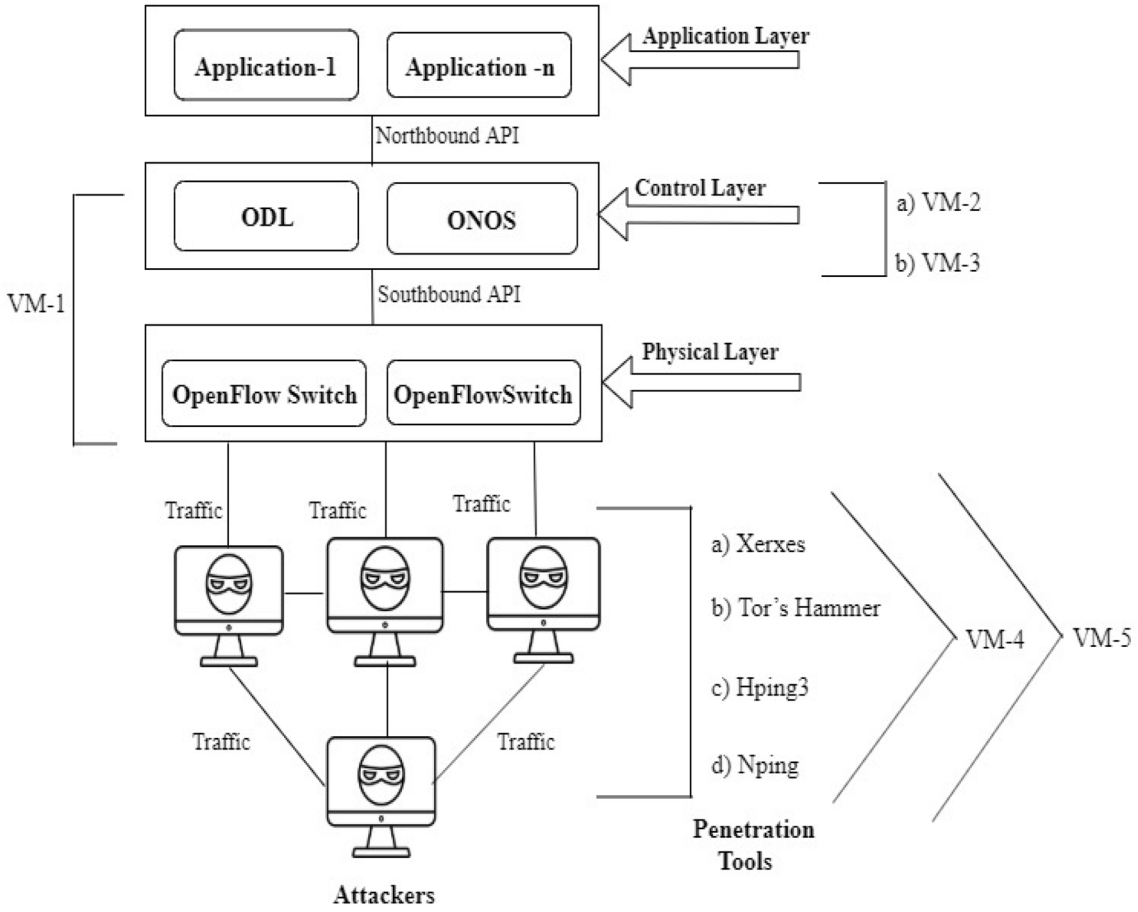


Fig. 1 Experimental setup

industry especially in the data center industry, where large numbers of servers are connected with the switches. SDN brings benefits like low-cost network infrastructure, faster implementation and troubleshooting of networks, better visibility of the networks, and adding programmability and customization in the networking [7, 8].

SDN is evolving rapidly in the network industry and according to Market research future, it is going to grow at CAGR of 42.41% which will make its market size from USD 8.82 billion in the year 2018 to USD59.9 billion [9] in year 2019. Another market research shows SDN to grow at CADR of 26.8% [10] from the year 2018 to 2023 with market size growing around USD 28.8 billion. SDN also gave rise to various new technologies SD-WAN, SD-Storage, 5G, etc. and is also integrated [11] with various new technologies like Cloud Computing, Intent-Based Networking, and network security [12]. SDN architecture is comprised of three layers i.e. Application Layer, Control Layer and Infrastructure Layer [13]. Application Layer is the topmost layer of the SDN Architecture Model and it includes the programs that communicate the behaviors and various resources by using Application Programming Interfaces (APIs) [14].

SDN Controller is the intermediate layer and it uses the information from the devices at the infrastructure layer and then also talks with the SDN applications with an abstract network view that includes events and statistics [15]. Infrastructures Layer controls the data plane work and takes the path based instructions from the controller and use that for processing and data forwarding [16]. SDN Controller is the main target for attackers as SDN is centralized and controls the network infrastructure by providing path related instructions to the data plane. Attackers try to get into the controller or by spoofing the controller [17].

If there occurs the compromisation of SDN controller, then the hacker can gain control over the network. SDN is vulnerable because of the decoupling of control and data plane as any wrong communication between the two planes can result in big loophole [17, 18]. In this article we have taken two widely adopted SDN controllers (ODL and ONOS) for our experimentation because they are practically being used by many big companies [19]. Although

Table 1 Different Machine's Specification

Name of the VM	IP addresses	Specifications
E Emulation tool Mininet (VM-1)	192.168.9.200	6 64-bit Ubuntu VM with 4 GB RAM (i7 processor)
3 3-node ODL clusters (VM-2)	192.168.9.208	6 32-bit Ubuntu VM with 4 GB RAM (i7 processor)
ONOS (VM-3)	192.168.9.203	64
		32-bit Ubuntu VM with 3 GB RAM (i7 processor)
Kalli- Linux (VM-4)	1 1192.168.1.6	64
		64 -bit Ubuntu VM with 2 GB RAM (i7 processor)
SNORT (VM-5)	192.168.9.201	64 32 -bit Ubuntu VM with 1 GB RAM (i7 processor)

Table 2 Results for varied scenarios and parameters used

Scenarios	Parameters							
	Controller	Number of packets/sec	Number of hosts and switches	Time in seconds when controller went Down	Type of network traffic	DDoS attack detection time in seconds	RTT in sec	Packet loss (%)
I	ODL	50,000	50	25	TCP SYN and HTTP	1	1097.6	97.9
	ONOS	50,000	50	23	TCP SYN and HTTP	2.3	1178	98
II	ODL	1,00,000	100	20	TCP SYN	2.1	119.8	99.8
	ONOS	1,00,000	100	17	TCP SYN	3	120.6	99.9
III	ODL	1,50,000	150	18	HTTP	4	0	100
	ONOS	1,50,000	150	15	HTTP	5.2	0	100
IV	ODL	2,00,000	200	14	TCP SYN	6	0	100
	ONOS	2,00,000	200	10	TCP SYN	7.5	0	100
V	ODL	2,50,000	250	11	TCP SYN and HTTP	8	0	100
	ONOS	2,50,000	250	08	TCP SYN and HTTP	10	0	100

SDN has got a large number of benefits at the same time it also suffers from some challenges as well such as security. There exists different varied types of network attacks on SDN such as security on data plane, IP spoofing, control plane,Man-in-the-Middle Attacks but Distributed Denial of Service Attacks (DDoS) attack is one of the most popular and disruptive attacks among all [20]. DDoS attacks on the controller can be used to disrupt the network services of the controller. These attacks try to exploit the bandwidth and scaling limits of SDN infrastructure. The first and primary step for DDoS attacks is to detect them on the network. Many researchers have contributed to the detection of DDoS attacks on the SDN network [21]. In this paper a DDoS detection system is created. We have integrated it with the two different SDN controllers. In order to analyse its performance we have considered different network scenarios and parameters. Our tool ensures the timely detection of the fast DDoS attacks.

1.1 Major contributions

(1) Related recent work is discussed. (2) As far as we are known we are the first to analyze the vulnerability of 3 node-ODL clusters and ONOS by using different penetration tools (Hping3, Nping, Xerxes, Tor Hammer, LOIC) from DDoS attacks (HTTP, TCP SYN) and thus implementing a detection tool by integrating SNORT IDS with them. (3) The experimentation is performed on the Mininet emulator tool in which five different VM's are created and connected with each other through a virtual switch. (4) SNORT IDS tool is integrated with ODL and ONOS to getting the alerts and thus creating a log file. (5) We have considered five different scenarios for the network traffic, evaluation and comparison is executed on the basis of parameters such as Number of data packets flooded, Round Trip Time (RTT), Time when the controllers went down, Type of DDoS attack, DDoS attack detection time, Number

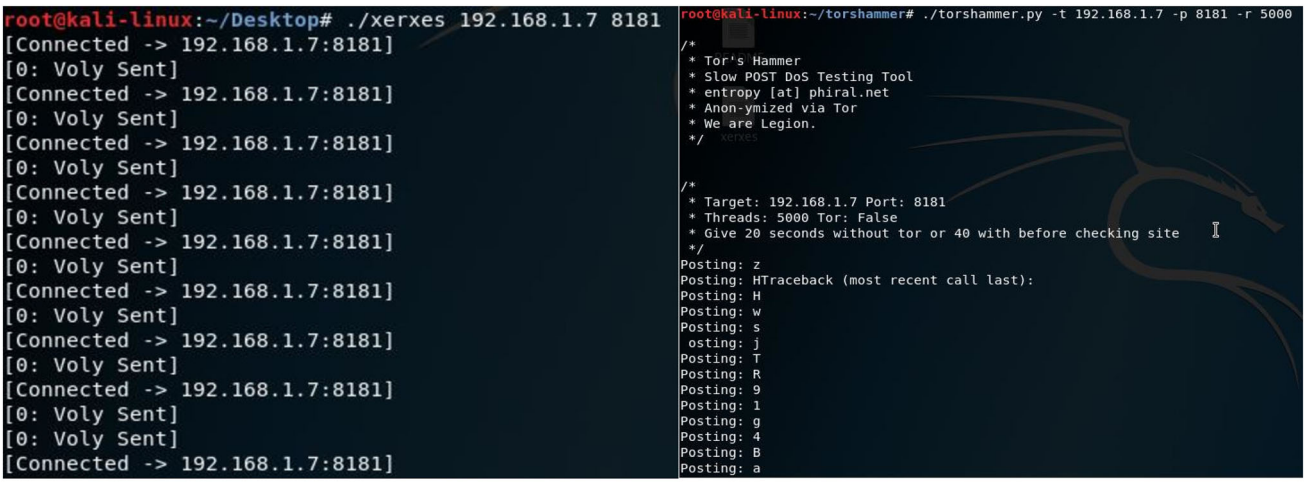


Fig. 2 Attacking controller with Xerxes and Tor's hammer

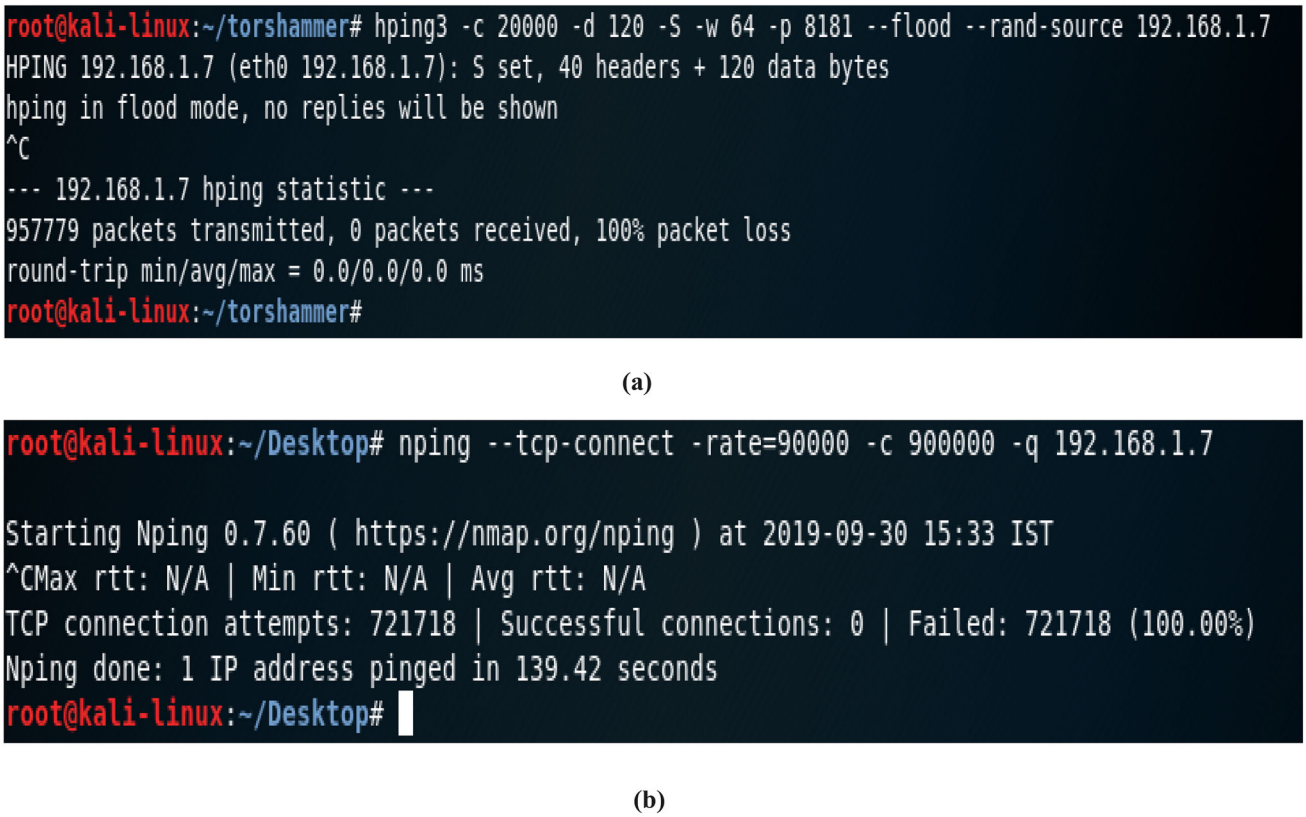
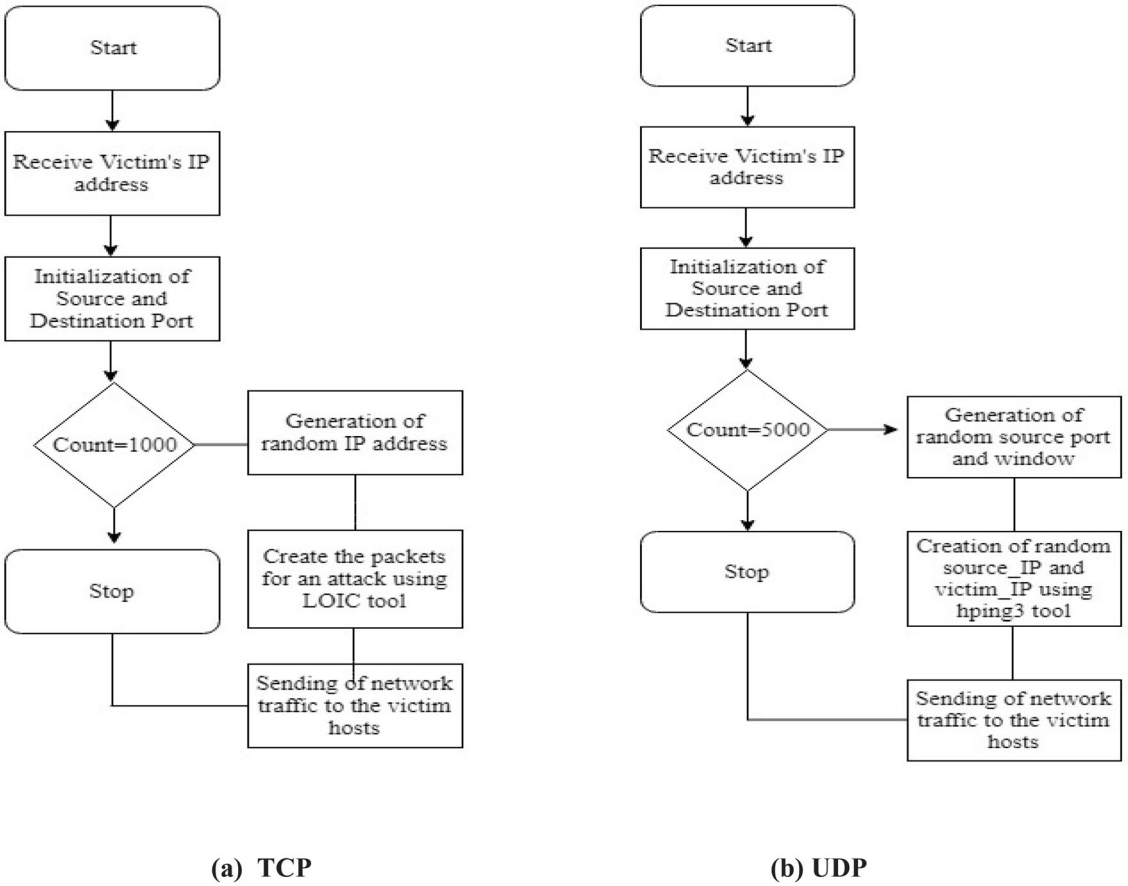


Fig. 3 Attacking controller with hping3 and Nping

of hosts and, Percentage of packet loss. (6) From experimentation we have found that our detection tool timely detects the DDoS attacks (HTTP and TCP SYN) with respect to the aforementioned parameters and different network scenarios.

1.2 Structure of the paper

The remainder of the paper can be categorized into various sections. Section 2 describes the related work. In Sect. 3 methodology used for the experimentation is illustrated



(a) TCP **(b) UDP**

Fig. 4 TCP and UDP pre-defined ports during experimentation

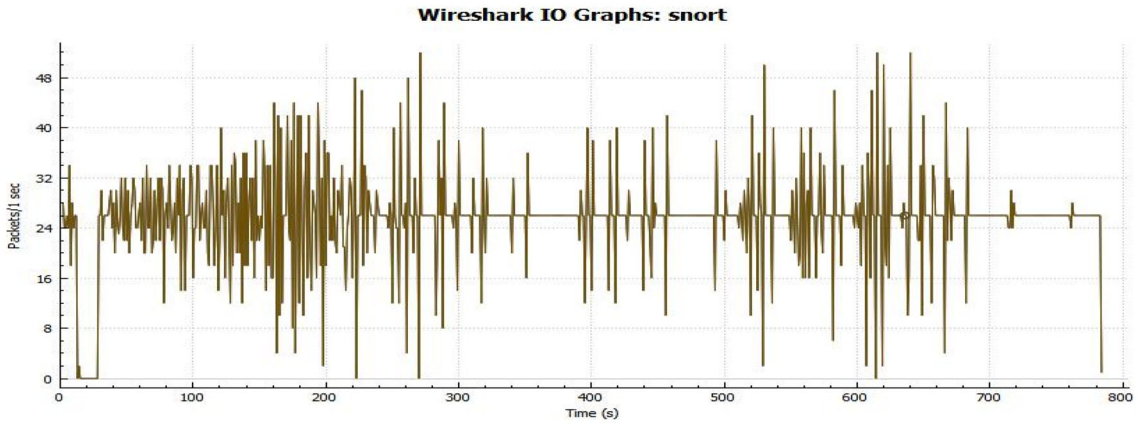


Fig. 5 Normal Traffic sent per second

Sect. 4 depicts the results drawn from the experimentation while in Sect. 5 discussion is provided. Finally in Sect. 6 conclusion of our work along with the future scope is given.

2 Related work

SDN controllers are the main core part of the entire SDN based network. The entire functionality is maintained by it only. Once the controller is down the whole network is automatically collapses. With an exponential popularity growth security in SDN is one of the important and crucial

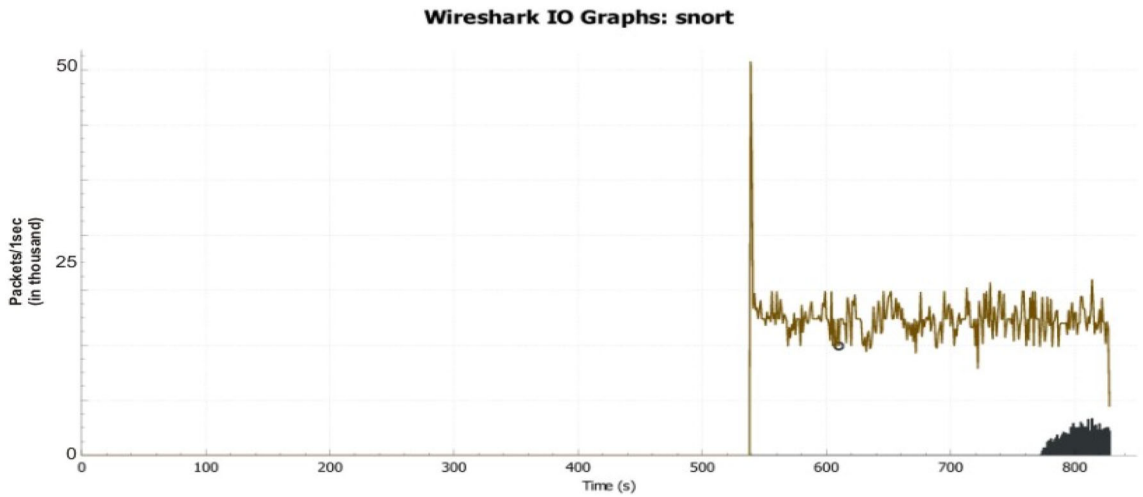


Fig. 6 TCP traffic flood for the first scenario up to 50,000 packets/s approximately

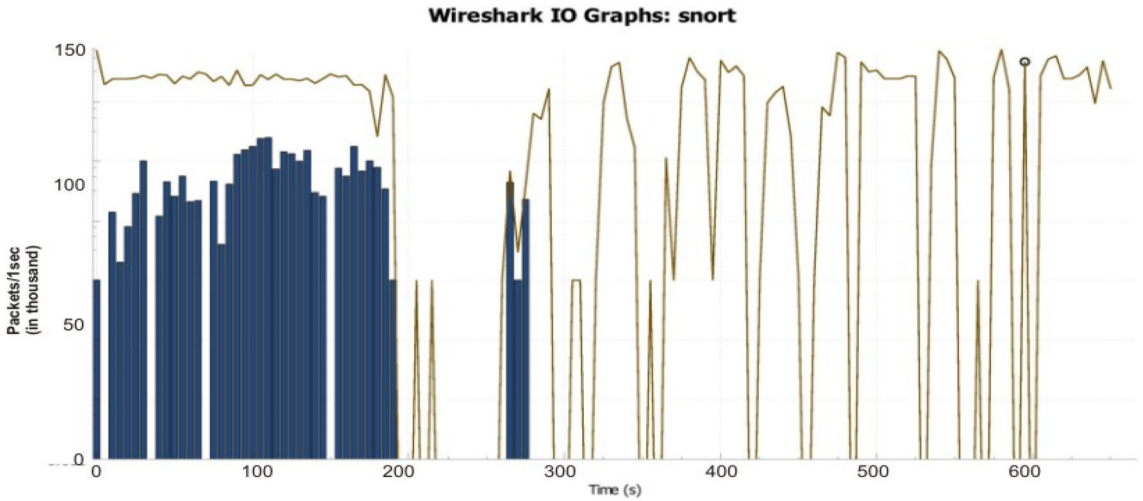


Fig. 7 TCP traffic flooded for the first scenario up to 1,50,000 packets/s approximately

tasks. Many intruders try to gain control over the SDN controller so that the global visibility and overall functionality can be achieved. By flooding huge traffic towards the controller, they may go down. Before the mitigation of DDoS attacks, the primary step is to detect them. Many authors have worked on either proposing or implementing such Intrusion Detection Systems (IDS). The most popular DDoS detection tool in this regard is Avant-Guard [22]. This model makes use of two different types of elements: First is the migration of the established link and second is the trigger of actuation. The first element possesses a specialization of the proxy that it receives.

According to this, it does the classification of multiple TCP-SYN requests. The second module creates an event to the SDN controller once the first element is done with the classification of the malicious network traffic. This work was oriented for only TCP-SYN attacks. There is the availability of many other solutions for detecting DDoS attacks as well. These solutions propose a method for the same by making use of bandwidth control. By configuring the bandwidth of each and every networking device (router) authors proposed the defense solutions [23]. Once this solution achieves the detection it also gives the notification to the SDN controller and makes amendments in the bandwidth. The limitation of their work was that the proposed system cannot distinguish the legitimate and unauthorized traffic in the network. Other researchers illustrate the collaboration between IDS and SDN [24].

In continuation of this, there are some other techniques as well which use the technique of signature-based detection method [25–32]. The disadvantage of their work was that these models or system cannot detect novel attacks or known attack variants. In [27] have proposed a model for early DDoS detection against SDN controllers. They have

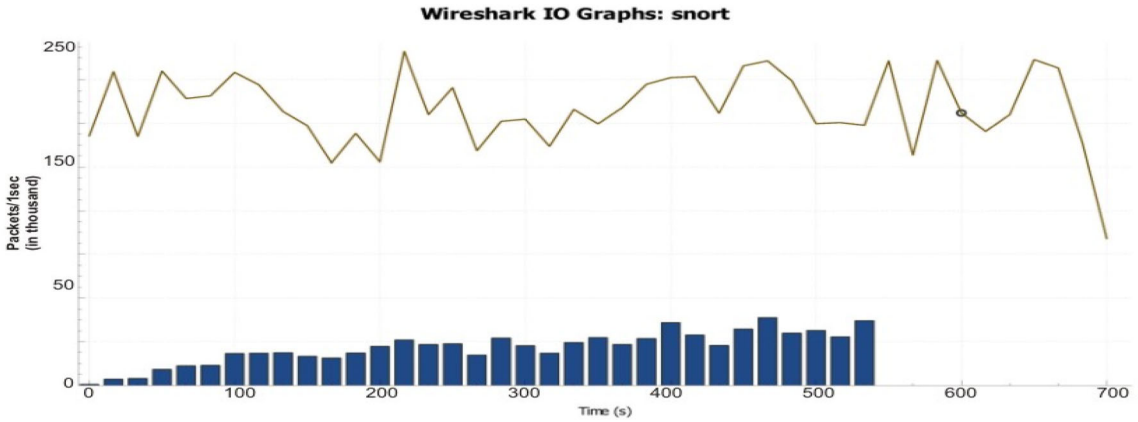


Fig. 8 TCP Traffic flood for the first scenario up to 2,50,000 packets/s approximately

```
root@onos-sdn:/etc/snort/rules# snort -T -i eth0 -c /etc/snort/snort.conf
Running in Test mode

       --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 80:81 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988 7000:7001 714
4:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9
999 11371 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
PortVar 'SSH_PORTS' defined :  [ 22 ]
PortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 3702 4343 4848 5250 6988
7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8243 8280 8300 8800 8888 8899 9000 9060 9080 909
0:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined :  [ 2123 2152 3386 ]
WARNING: /etc/snort/snort.conf(115) Var 'RULE_PATH' redefined
```

Fig. 9 Configuration and testing mode in SNORT

made use of entropy to detect the DDoS attack. But authors were limited to the number of packets flooded and for multi-controller an environment like 3-node ODL cluster their experimentation was not executed. Similarly, in [33] authors proposed another method for early DDoS detection. Authors were limited to very few parameters used for their experimentation and they had used a previously existing data set to find out the accuracy of their proposed model. Making use of machine learning for DDoS detection is also done in SDN by many authors. In [34] authors have proposed an early DDoS detection and prevention method using SNORT. They have made use of the Ryu SDN controller for their experimentation but authors were limited to the parameters used, tools for penetration testing, number of packets flooded a type of DDoS attack. Recently another entropy-based DDoS detection tool was proposed by Ahalawat et al. in [35] but their work was limited to only the UDP type of DDoS attack. Therefore, from the existing work related to DDoS detection tool, it can be concluded that available tools or system are either limited

```
GNU nano 2.2.6                          File: /etc/snort/rules/local.rules

alert tcp any any -> $HOME_NET 8181 (msg:"SDN connection attempt"; sid:1000001; rev:1;)
alert icmp any any -> $HOME_NET any (msg:"ICMP connection attempt"; sid:1000002; rev:1;)
alert tcp any any -> $HOME_NET 80 (msg:"Web connection attempt"; sid:1000003; rev:1;)
alert tcp any any -> $HOME_NET 22 (msg:"SSH connection attempt"; sid:1000004; rev:1;)
```

Fig. 10 Local rules created in SNORT

```
root@onos-sdn:/etc/snort/rules# snort -A console -q -c /etc/snort/snort.conf -i eth0
09/02-14:57:29.088661  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.089019  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.091655  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.092518  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.123790  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.124971  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.167203  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.174551  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.589121  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.596786  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.596802  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.596806  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.596811  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.596815  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.596818  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.600206  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.600243  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.601977  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.601990  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.601995  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.601999  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
09/02-14:57:29.609073  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.626827  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.626847  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.626856  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.626860  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.626864  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52652 -> 192.168.1.36:8181
09/02-14:57:29.650938  [**] [1:1000001:1] SDN connection attempt [**] [Priority: 0] {TCP} 192.168.1.6:52651 -> 192.168.1.36:8181
```

Fig. 11 Alerts in SNORT

to parameters, penetration tools, type of DDoS attacks, no support for multi-controller (like 3-node ODL cluster).

3 Methodology

In this section methodology which is used to carry out the experimentation is discussed. Various tools and scenarios are discussed in detailed.

Mininet Emulation tool In order to detect various DDoS attacks against the centralized controller, a network emulator tool, Mininet [36, 37] is being used. For the creation of a virtual network, this tool proves to be very useful. This tool helps to create the number of hosts and switches. It creates the OpenFlow Switches [38] for various versions like 1.0, 1.2, 1.3, etc. to provide high flexibility customized routing in SDN. It creates a network that comprises a virtual network of hosts, switches, and controllers along with a secure communication link among them. For our experimentation OpenFlow protocol version, 1.3 is used.

VM's and SNORT We have created five different VM's, VM-1, VM-2, VM-3, VM-4, and VM-5. VM-1 is comprised of Mininet as shown in Fig. 1. A 3-node cluster of ODL (Beryllium version) is providing the multi-controller environment for experimentation. ODL controller has a very large platform with a lot of plugins and features are created in VM-2. VM-3 includes the ONOS machine (Peacock version). The VM-2 is integrated with the SNORT, which is an open-source network intrusion prevention system. VM-4 is comprised of Kalli Linux which further includes different penetration tools to launch successful DDoS attacks. Four different penetration tools are used to launch the DDoS attacks described below in Sect. 3.1. SNORT [38, 39] is capable of performing real-time traffic analysis and packet logging on IP networks created in VM-5. Analyzation of various protocols, searching/matching of the data, and detection of the variety of attacks and probes can be performed by it [40]. SNORT is used to get the alerts for the DDoS detection and a dataset is created by collecting the data from the SNORT IDS. The different machines having varied hardware specifications used in experimentation are shown in Table 1. For our experimentation, we have used a data-

centric tree topology having a different number of hosts and switches for different scenarios.

3.1 Penetration tools used to launch DDoS attacks

In our experimentation different Opensource DDoS penetration tools are used to first check the vulnerability of both the controllers from these attacks and then for comparison in varied network scenarios having different parameters. These tools are described as follows:

Xerxes It is a DoS tool that works in the most efficient manner. It is developed by hackers to alter DoS attacks. To launch the various independent attacks against many target web-sites while not essentially requiring a botnet. For our experimentation purpose, we have used Kali Linux and implemented the added Xerxes code from the repository of the Xerxes tool [41].

Tor's Hammer It is another type of DoS tool which is created by phiral.net for slow-rate hypertext transfer protocol POST (Layer 7). It carries out a DoS attack by employing a classic slow POST attack, wherever hypertext mark-up language POST field's square measure transmitted in slow rates underneath a similar session (actual rates square measure at randomly chosen at intervals the limit of 0.5–3 s) [42].

Hping3 Hping3 is a command-line oriented tool and analyzer for various TCP/IP packets. The interface is affected by the ping (8) software package command, but hping isn't exclusively able to send ICMP echo requests. It supports the UDP, ICMP and RAW-IP protocols contain a traceroute mode, the facility to send files between a secure communication line, and lots of other choices [43].

Nping It is also another widely used DoS tool. It is an open-supply tool for network packet generation, response analysis, and latency activity. Nping can generate network packets for a decent varies of protocols, allowing users full management over protocol headers. It usually used as a straightforward ping utility to sight active hosts, it can also be used as a raw packet generator for network stack stress testing, poisoning in ARP, DoS attacks and route tracing, etc. [44].

3.2 Network scenarios and parameters used for evaluation and detection

As mentioned earlier there are different parameters used for different scenarios in our experimentation setup. The range in these parameters changes with respect to every scenario. These parameters (with changes in the range) and a different scenario is described as follows:

3.2.1 Parameters used for evaluation

The number *of data packets* flooded towards the controllers from different penetration tools. In every different network scenario, these are increased by 50,000 packets/s. While measuring this parameter, there is one another type of parameter to get evaluated i.e. *type of DDoS attack.* From different penetration tools, different types of attacks are generated. In our experimentation, we have used HTTP based flood attacks and TCP SYN attacks. Once the successful launch of the DDoS attack occurs there is a need to know the *time when the SDN controllers went down*. Analyzing the time at which the SDN controllers went down is one of the vital parameters for or evaluation. After the controllers are down with a huge amount of traffic, the time taken by the SNORT IDS to get the alerts is a crucial parameter. *RTT* among hosts is analyzed for all the scenarios. These numbers of hosts are increased by 50 hosts in every network scenario. *Percentage of packet loss* in different scenarios having different *numbers of hosts, type of traffic* is also analyzed.

3.2.2 Network scenarios

First of all, by making use of the Mininet emulation tool, data-centric tree topology is created. There are five different network scenarios used in our experimentation. In Scenario I traffic having 50 numbers of hosts, 50,000 packets/s bombarded towards the controller. In scenario II number of hosts is increased to 100 and 100,000 packets/second are bombarded. Similarly, in Scenario III, IV, and V, the number of hosts increased from 150, 200, 250 whereas flooded traffic is increased by 1,50,000 packets/s, 2, 00,000 packets/s and 2, 50,000 packets per second respectively.

4 Results of the experimentation

In Linux based VM all the simulations and experiments were performed. Kali Linux and SDN controller both are in the same network and having IP addresses i.e. 192.168.1.6 and 192.168.1.7. During experimentation, we have used HTTP flood attacks and TCP SYN attacks. From the kalli Linux VM, these different penetration tools were run and successfully penetrated the DDoS attacks on the VM-2 and VM-3.Xerxes and Tor Hammer are used for HTTP Flood attacks on port number 8181, while on the other hand Hping3 and Nping are used for TCP SYN attacks. Kali Linux and SDN controllers both are in the same network and have IP addresses i.e. 192.168.9.208 and 192.168.9.203.

4.1 Integration of different penetration tools with the controllers

Below Fig. 2a shows the Xerxes tool and in Fig. 2b hammer tool used for DDoS attack. On the same Kali Machine, we are running both DDoS tools. As ODL and ONOS use TCP port number 8181 for HTTP, therefore we have used port 8181 in our attack command. Another tool Hping3 and Nping to flood the traffic towards ODL and ONOS as shown in Fig. 3a and b.

We have used Hping3 and Nping for TCP SYN Flood DDoS attack and below are commands used for Hping3 to flood TCP SYN traffic on ODL and ONOS. In Hping3, we have used, following attributes:-c—Number of packets to send. –S—SYN Packets. –p—Port Number. –flood—To flood the traffic. –rand-source—Using Random IP Addresses to attack. On Nping, we are using following attributes: -tcp-connect—It is unprivileged TCP Connect Probe Mode. –rate—Send number of packets per second. -c—Stop after < n > of rounds. –q—Decrease verbosity level by one.

During traffic generation in a TCP SYN flooding DDoS attack, IP addresses, port number of the target along with the number of packets must be calculated and known. After this, a new IP packet having a random source IP and the target's IP will be setup. Along with this, there has to be the creation of TCP packet having a random source port, which is further comprising the victim port's flag, sequence in various packets, and window time for the multiple packets. These multiple IP Packets will be bombarded to the victim host. The TCP pre-defined ports during experimentation on the SDN network can be clearly seen in can be seen in Fig. 4a. Each time, different IP packets are being generated and hping 3 tool is used to achieve this huge amount of traffic.

On the other hand, in case of traffic generation of random UDP packets, the victim host is bombarded with multiple random UDP packets. During this process of traffic generation towards the victim, first of all, the IP addresses of the target are determined; once it is achieved both the ports (source and destination) are initialized to 80 and 1. Every time, varied IP packets are generated. For our experimentation, we have set a predefined port count of 1000. The Low Orbit Ion Cannon (LOIC) is an open-source network stress testing and DoS attack application, written in C sharp language is used to achieve this. Once the IP packets are created, they are to be sent within the given time interval towards the IP address victim. On the SDN network, a detailed process and various steps for the UDP flooding attack can be seen in Fig. 4b).

4.2 Traffic generation

Before the penetration of DDoS attacks from different DDoS attacks the normal traffic, the rate was 5000 packets/second. Figure 5 is showing the rate of normal traffic. This includes the legitimate traffic sent per second. It can be clearly seen that the highest rate of normal traffic sent was around 5000 packets/second. These log files were captured using Wireshark from the real-time traffic in the network. Once the traffic starts flowing in the network, from different penetration tools as mentioned in Fig. 2 and 3. The huge amount of TCP and HTTP traffic is bombarded towards the controllers. Figures 6, 7 and 8 are illustrating the log files for the same. In Fig. 6 the network is bombarded with up to 50,000 packets/second, it is the scenario I whereas in Fig. 7 the number of packets was increased to 1, 50,000 packets/s. It can be clearly seen the deviation of normal traffic and TCP/ HTTP errors. The gray line in the Figures depicts the normal traffic whereas blue bars are depicting the TCP and HTTP error.

4.3 Integration with the SNORT

Once the SNORT is Configure and is running in test mode. By default, with pre-defined ports shown in the SNORT initialization as shown in below Fig. 9. After the successful launch of DDoS attacks from the various aforementioned penetration tools around 2,50,000 packets per second were bombarded towards the SDN controllers in different scenarios and ultimately make them down and unavailable for any functionality. Once the controllers are down in SNORT, rules can be created inside the /etc./SNORT/rules directory under local.rules file, where we can add alert types on the basis of traffic coming on our service or applications.

As we are using ODL and ONOS controllers and both use TCP port no 8181 as HTTP ports and it can be attacked for both HTTP and TCP SYN Flood attack. As mentioned earlier as well that we have tested different DDoS attacks on these controllers that clearly take down the controller after a successful attack. Different SNORT rules can be used for the detection of DDoS attacks by configuring SDN DDoS alert rules in local rules. We have Configured alert rules by configuring source traffic from any network or any port and if that is coming on the SDN controller at TCP Port Number 8181, then the message can be listed as an SDN connection attempt from an outside network which is not config. The local rules created are shown below in Fig. 10. After creating the rules in local.rules section, we have attempted a DDoS attack using various tools and snort console shows the alerts on incoming traffic on SDN Controller over Port number 8181 alerts are generated as

shown in Fig. 11 below in which the SDN connection attempts from 192.168.1.6.

In the above-mentioned Table 2, it is clearly showing the different parameters used for multiple scenarios. The detection time when the number of packets was 50,000 in the I scenario is 1 s and it is increased to 10 s against the 2, 50,000 flooded network packets. The time when the controllers went down and the functionality of the overall network is also stopped is dependent upon the number of packets bombarded. Whereas for the scenario's I and II the packet loss is 97–99.9%, it is observed that for the scenarios IV and V the packet loss comes out to be 100%. The type of traffic for both the controllers in these scenarios is same.

5 Discussion

Before the DDoS attacks, the normal rate of traffic was around 5000 packets/ second as shown in Fig. 5. From the experimentation performed and illustrated in Figs. 2, 3, 6, 7 with the help of different penetration tools there occurs the flooding of huge (TCP SYN and HTTP) network traffic towards the SDN controllers. It can be seen that both the SDN controllers ODL (3 node cluster) and ONOS are vulnerable to DDoS attacks. For the different scenarios and parameters used as illustrated in Table 2, it is observed that as the rate of network traffic is increased the DDoS detection time also increases. The maximum amount of traffic flooded towards the controllers was 2,50,000 packets/s and the detection time corresponding to that was 8 s, 10 s for ODL and ONOS respectively. While the DDoS attacks were being detected we have also taken care of other vital parameters involved as well. In the first scenario, only 50,000 packets were flooded and a number of hosts, switches taken in the topology were 50, it is observed that 97.9% packets are lost but with the gradual increment in the number of hosts and traffic flood the packets loss came down at 100%. This depicts that the DDoS detection time is directly proportional to the number of network devices (hosts, switches) used and the amount of traffic bombarded. RTT was also decreased from 1097.6 s to 0 with the rate of increase in the number of packets flooded and number of hosts/ switches. When SNORT was Configure and rules were created in order to get the alerts for the DDoS attack detection shown in Fig. 9, 10 and 11. Therefore, it can be concluded that DDoS detection is performed in its early stages and suitable actions can be taken after wards.

6 Conclusion and future scope

As SDN is gaining popularity day by day, its demand is also increasing. But with the multiple advantages there exists numeous challenges as well. One of the challenge is security in SDN. The isolation of control plane from the data plane becomes a vulnerable point for the intruders. DDoS attack in SDN is considered as one of the most prominent attack.

Before the network administrator can mitigate it, the first step is to detect them. In this paper a DDoS detection system is implemented by using SNORT IDS (Intrusion Detection System) on the ODL and ONOS. In order to analyse the performance of the implemented DDoS detection tool, multiple scenarios having varied number of hosts, switches and generated data traffic are used.For traffic generation different penetration tools were used wheres on the on the other hand for varied number of hosts/ switches, the Mininet emulation tool is used. The evaluation of the DDoS detection tool was achieved on the basis of selected parameters. It is found that out of both the controllers ODL takes less time to detect DDoS attack and goes down in later time than ONOS. There are many researchers currently working in the security of SDN. But to our best knowledge, no researcher has made use of SNORT IDS with ODL and ONOS. This paper will be helpful in giving a new direction for the researchers. In the future, a DDoS prevention framework can be a study point.

References

1. Xia, W., Wen, Y., Foh, C.H., Niyato, D., Xie, H.: A survey on software-defined networking. IEEE Commun. Surveys Tutor. **17**(1), 27–51 (2015)
2. Lantz, B., Heller, B., & McKeown, N.: A network in a laptop: rapid prototyping for software-defined networks. In Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, p. 19. ACM (2010)
3. Feamster, N., Rexford, J., Zegura, E.: The road to SDN: an intellectual history of programmable networks. ACM SIGCOMM Comput. Commun. Rev. **44**(2), 87–98 (2014)
4. Nunes, B.A.A., Mendonca, M., Nguyen, X.N., Obraczka, K., Turletti, T.: A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. **16**(3), 1617–1634 (2014)
5. Shenker, S., Casado, M., Koponen, T., McKeown, N.: The future of networking, and the past of protocols. Open Netw. Summit **20**, 1–30 (2011)
6. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Turner, J.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
7. Fernandez, M.P.: Comparing OpenFlow controller paradigms scalability: reactive and proactive. In: 2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 1009–1016. IEEE (2013)

8. Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
9. https://www.globenewswire.com/newsrelease/2019/04/04/1797303/0/en/Software-Defined-Networking-SDN-Market-Size-USD-59-Billion-by-202Growing-at-Massive-CAGR-of-42-41.html. Accessed 01 Oct 2019
10. https://www.transformingnetworkinfrastructure.com/news/2019/03/22/8923883.htm. Accessed 15 Oct 2019
11. https://www.networkworld.com/article/3209131/what-sdn-is-and-where-its-going.html. Accessed 30 Oct 2019
12. Gupta, B.B., Agrawal, D.P. (eds.): Handbook of Research on Cloud Computing and Big Data Applications in IoT. IGI Global, Pennsylvania (2019)
13. Jammal, M., Singh, T., Shami, A., Asal, R., Li, Y.: Software defined networking: state of the art and research challenges. Comput. Netw. **72**, 74–98 (2014)
14. Badotra, S., Singh, J.: A review paper on software defined networking. Int. J. Adv. Res. Comput. Sci. **8**(3), 2 (2017)
15. Kamal, A.E., Han, L., Lu, L., Jabbar, S.: Guest editorial: Special issue on software defined networking: trends, challenges, and prospective smart solutions. Peer-to-Peer Netw. Appl. **12**(2), 291–294 (2019)
16. Nayyer, A., Sharma, A.K., Awasthi, L.K.: Issues in software-defined networking. In: Proceedings of 2nd International Conference on Communication, Computing and Networking, pp. 989–997. Springer, Singapore (2019)
17. Bhushan, K., Gupta, B.B.: Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. J. Amb. Intell. Hum. Comput. **10**(5), 1985–1997 (2019)
18. Scott-Hayward, S., O'Callaghan, G., Sezer, S.: SDN security: A survey. In 2013 IEEE SDN For Future Networks and Services (SDN4FNS), pp. 1–7. IEEE (2013)
19. Badotra, S., & Panda, S.N. Evaluation and comparison of OpenDayLight and open networking operating system in software-defined networking. Cluster Computing, pp. 1–11
20. Fernandes, G., Rodrigues, J.J., Carvalho, L.F., Al-Muhtadi, J.F., Proença, M.L.: A comprehensive survey on network anomaly detection. Telecommun. Syst. **70**(3), 447–489 (2019)
21. Gupta, B.B., Badve, O.P.: Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. Neural Comput. Appl. **28**(12), 3655–3682 (2017)
22. Shin, S., Yegneswaran, V., Porras, P., Gu, G.: Avant-guard: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 413–424. ACM (2013)
23. Piedrahita, A.F.M., Rueda, S., Mattos, D.M., & Duarte, O.C.M.: FlowFence: a denial of service defense system for software defined networking. In: 2015 Global Information Infrastructure and Networking Symposium (GIIS), pp. 1–6. IEEE (2015)
24. Ombase, P.M., Kulkarni, N.P., Bagade, S.T., Mhaisgawali, A.V.: DoS attack mitigation using rule based and anomaly based techniques in software defined networking. In: 2017 International Conference on Inventive Computing and Informatics (ICICI), pp. 469–475. IEEE (2017)
25. You, X., Feng, Y., Sakurai, K.: Packet In message based DDoS attack detection in SDN network using OpenFlow. In: 2017 Fifth International Symposium on Computing and Networking (CANDAR), pp. 522–528. IEEE (2017)
26. Kia, M. (2015). Early Detection and Mitigation of DDoS Attacks in Software Defined Networks (Doctoral dissertation, Master's Thesis, Ryerson University, Toronto, ON, Canada).
27. Mousavi, S.M., & St-Hilaire, M.: Early detection of DDoS attacks against SDN controllers. In: 2015 International Conference on Computing, Networking and Communications (ICNC), pp. 77–81. IEEE (2015)
28. Xing, T., Huang, D., Xu, L., Chung, C.J., Khatkar, P.: Snortflow: Aopenflow-based intrusion prevention system in cloud environment. In: 2013 second GENI research and educational experiment workshop, pp. 89–92. IEEE (2013)
29. Sahay, R., Blanc, G., Zhang, Z., Debar, H.: Towards autonomic DDoS mitigation using software defined networking (2015)
30. Chowdhary, A., Pisharody, S., Alshamrani, A., Huang, D.: Dynamic game based security framework in SDN-enabled cloud networking environments. In: Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, pp. 53–58. ACM (2017)
31. Jevtic, S., Lotfalizadeh, H., Kim, D.S.: Toward network-based ddos detection in software-defined networks. In: Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication, p. 40. ACM (2018)
32. Choi, Y.: Implementation of content-oriented networking architecture (CONA): a focus on DDoS countermeasure. In: Proceedings of European NetFPGA developers workshop (2010)
33. Wang, R., Jia, Z., Ju, L.: An entropy-based distributed DDoS detection mechanism in software-defined networking. In: 2015 IEEE Trustcom/BigDataSE/ISPA, Vol. 1, pp. 310–317. IEEE (2015)
34. Manso, P., Moura, J., Serrão, C.: SDN-based intrusion detection system for early detection and mitigation of DDoS attacks. Information **10**(3), 106 (2019)
35. Ahalawat, A., Dash, S.S., Panda, A., Babu, K.S.: Entropy Based DDoS Detection and Mitigation in OpenFlow Enabled SDN. In: 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), pp. 1–5. IEEE (2019)
36. www.mininet.org. Accessed 02 Dec 2019
37. Badotra, S., Singh, J.: Open daylight as a controller for software defined networking. Int. J. Adv. Res. Comput. Sci. **8**(5), 1105–1111 (2017)
38. https://snort.org/. Accessed 28 Dec 2019
39. Roesch, M.: Snort: lightweight intrusion detection for networks. Lisa **99**(1), 229–238 (1999)
40. Gupta, A., Sharma, L.S.: Performance evaluation of snort and suricata intrusion detection systems on ubuntu server. In: Proceedings of ICRIC 2019, pp. 811–821. Springer, Cham (2020)
41. Shorey, T., Subbaiah, D., Goyal, A., Sakxena, A., Mishra, A. K.: Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools. In: 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 318–322. IEEE (2018)
42. Behal, S., Kumar, K.: Characterization and comparison of DDoS attack tools and traffic generators: a review. IJ Netw. Security **19**(3), 383–393 (2017)
43. Hoque, N., Bhuyan, M.H., Baishya, R.C., Bhattacharyya, D.K., Kalita, J.K.: Network attacks: taxonomy, tools and systems. J. Netw. Comput. Appl. **40**, 307–324 (2014)
44. Guozi, S.U.N., Jiang, W., Yu, G.U., Danni, R.E.N., Huakang, L.I.: DDoS attacks and flash event detection based on flow characteristics in SDN. In: 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), pp. 1–6 (2018)

Sumit Badotra is currently pursuing Ph.D. in Computer Science and Engineering from Chitkara University, Punjab, India. He has completed his M. tech Computer Science and Engineering from MRSPTU, Bathinda, and Punjab, India and B. tech from IGPTU, Punjab, India. His research areas are Software Defined Networking, IoT and Network security.

Surya Narayan Panda has completed his Ph.D. (Computer Sci. & Appl.) from Kurukshetra University Kurukshetra, Haryana (India), M.Sc. (Computer Science) from Kurukshetra University, Kurukshetra, Haryana (India), B.Sc. (Hons) Distinction from Sambalpur University, Orissa (India), PGDEDP from National Institute of Electronics & IT formerly known as Regional Computer Centre, Chandigarh (India). He is currently working as Director (Research), Centre of Advanced Computing & Research, Chitkara University, Punjab Campus, and Rajpura Punjab, India. His domain of research are Technology Trends, Machine Vision, Communication, Security, Big Data, Bioinformatics, Cloud Computing, Communication, Communication Protocols, Computational Creativity, Computer Networks, Computer Vision, Cyber Security, Data Science, Databases, Decision Support Systems, Distributed Computing, e-Business, e-Commerce, e-Governance, Electronic Data Interchange (EDI), Internet of Things, Internet Security, Intrusion Detection, Machine Vision, Network Security.