DDOS Attack Detection & Prevention in SDN using OpenFlow Statistics

Nisha Ahuja[1], Gaurav Singal[2]
Department of CSE, *Bennett University*, Greater Noida, India
na6742@bennett.edu.in[1], gauravsingal789@gmail.com[2]

***Abstract*—Software defined Network is a network defined by software, which is one of the important feature which makes the legacy old networks to be flexible for dynamic configuration and so can cater to today's dynamic application requirement. It is a programmable network but it is prone to different type of attacks due to its centralized architecture. The author provided a solution to detect and prevent Distributed Denial of service attack in the paper. Mininet [5] which is a popular emulator for Software defined Network is used. We followed the approach in which collection of the traffic statistics from the various switches is done. After collection we calculated the packet rate and bandwidth which shoots up to high values when attack take place. The abrupt increase detects the attack which is then prevented by changing the forwarding logic of the host nodes to drop the packets instead of forwarding. After this, no more packets will be forwarded and then we also delete the forwarding rule in the flow table. Hence, we are finding out the change in packet rate and bandwidth to detect the attack and to prevent the attack we modify the forwarding logic of the switch flow table to drop the packets coming from malicious host instead of forwarding it.**

***Index Terms*—SDN, Mininet, Network attack, Traffic simulation, DDOS**

I. Introduction

A new network which is programmable is Software-Defined Network [1] is unlike the traditional network. It is also known as next-generation network. Software-Defined Network is a breakthrough in networking as it breaks the year old barriers of rigidity in network configurations. With the advent of SDN the network can be programmed at run time as per the user requirement. Open Networking Foundation(ONF) [1] defines SDN as the network in which there are some architectural differences from the traditional network in the sense SDN defines its working in two different planes. Data and control plane where the control plane is the brain of the network. With SDN there is a global vision of the network i.e. the status of all the switches existing in the topology is known so the rules for traffic routing can be decided intelligently. SDN is invented as a brainchild of a Ph.D Student Martin Casado during his thesis writing in 2005 at Stanford University. In nutshell we can say that Software-defined network is a network which is defined and managed by software. Also, when combined with the concept of Network function virtualization (NFV) [1] it brings out virtualization of entire networking hardware.

The traditional network cannot be programmed by the software. The devices in the network like switch, router are hard to configure as the two planes in the device i.e. Data Plane and Control Plane are tightly coupled. It is basically due to the distributed nature of traditional network and thus inflexible [1]. But Software-Defined Network use the concept of splitting the network into data and Control Plane. Control Plane [2] is the one responsible for making all the decisions in the network. We can see the control plane in SDN like the brain in human body and Data Plane is known for its functionality of just forwarding the packets.

This architectural change has brought a lot of flexibility to the network. In nutshell, traditional network routing and forwarding decisions are made by the switches based on the destination IP address which is unlike Software Defined Network. In SDN the routing and forwarding decisions [3] are decoupled and here traffic forwarding is based on flow-based scheme. A flow is defined as a rule based on the match which is then used to forward the traffic.

Various companies have also started looking for the option of investing in SDN. One of the big industry giants like Google is using SDN in their data centers. Firms like NEC to Hewlett-Packard have setup a SDN network. SDN consists of folllowing layers namely a) Forwarding layer consist of dumb switches b) Control layer consist of Controller c) Application layer consisting of application programs which the user can make to interact with the controller. These three layers communicate using Northbound and Southbound APIs [1]. A northbound interface is defined as the set of API's [1]between the upper level planes i.e. application plane and the control plane whereas southbound interface is the set of API's between the controller and the device plane.

Our motivation to work on DDOS attack is because the work which has been done till now is mostly focused on the idea of finding the randomness [5] in the traffic, based on Shannon's theory. But the proposed work here involve calculation at switches which is less time-consuming. DDOS attack can result in the following scenarios:

- Exhausting the memory of switch and controller.
- Exhausting the control channel bandwidth.
- Exhausting the computational power at data and control plane.

978-1-7281-4392-7/19/$31.00 ©2019 IEEE

Our contribution in this paper is to detect and prevent the DDOS attack by using Open Flow statistics which are extracted from the switches. Continuous monitoring of the switch is done and whenever the value of the packet rate is found to be more than '100' which is a already set threshold value, the attack is detected. The prevention method is also used in which the controller modify the flow entry from forward packets to drop the packets coming from the identified malicious host.

The paper is organized as outlined below, Section II detailed on the literature survey whereas Section III illustrates the method used for detecion and pevention of attack. and algorithm strategy. Section IV provides the Experimental Setup and Results and finally, Section V concludes the paper.

II. Related Work

Here we discuss the various state-of-the-art in the field of security [4] of the network specially in the context of DDOS attack. Security techniques [6] which have been discussed does not necessarily pertains to SDN, we have diuscussed the core techniques only, it is not necessary that they are SDN specific.

Hong et.al. [7] discuss the concept of mitigation of slow DDOS attack on the server by following a technique employed at the server. The algorithm for detection and mitigation is running on Controller. A Tcp connection has to be already established between SHDA (Secure hash Decryptable Analysis) and web server. Now, whenever attacker tries to connect to the web server it will send large amount of connection request $half - openconnections$ and when the count reaches a threshold, the attack is detected. When detection process running in the server detects the attack, it forwards the half-open connection packets to the mitigation algorithm running on the controller which will continue receiving of the half-open connections. After waiting for the stipulated time in which half-open connection of slow client will complete but the half-open connection of attacker will never complete. After waiting for a threshold limit of time the traffic is classified as attacked traffic or benign traffic. Once the controller add the IP's suspected into the blocked list, the attack is mitigated.

Kalkan et.al. [9] describes the concept of using entropy together with the attributes of TCP layer. For example the randomness in the destination IP address will not solely give accurate information of the attacker but some other parameters like Protocol type, TCP flags, destination port, source port, destination IP, Source IP and packet size are also important to consider for detecting the attack. It creates pairing of traffic features mentioned above and calculates the joint entropy (for example: Destination IP with TTL i.e. time to live) when in non-attacked position and compare it with the same pair after attack. If the difference in entropy values for the normal and attacked traffic exceeds the threshold value then the attack is detected and mitigation module start executing in which detected pairs during the previous phase are taken in ratio with the same during normal traffic flow and a ratio is calculated.If the value of the ratio comes to be greater than the set limit value then it starts dropping packets for that pair unless the bandwidth decreases to acceptable limits.

Da Silva et.al. [10] detailed on the architecture for detecting, classifying and preventing the attack in SDN. The author used the concept of Shannon's theory for the purpose of attack detection in two phases: In the first phase, illegal attack traffic is found by using entropy-based approach and in second phase the dump flows analysis and its classification is done using various machine learning classifiers which uses the past information collected to classify the abnormal traffic.

NisharaniMeti et.al. [11] proposed the use of various Machine-Learning algorithms to detect the attack in Software-Defined Network (SDN). Various Machine learning algorithms are tested for their performance in classification task. The dataset used is a TCP traffic set between certain locations obtained from the experimental results. With the help of controller the detection and prevention take place.Controller keeps an access contro list (ACL) with the help of which it segregates the TCP traffic set into normal or malicious traffic and then the labelled traffic set is used by the machine learning classifiers and it is found that SVM algorithm gives better results as compared to other machine learning algorihms.

Rasool et.al. [12] discuss the use of deep learning based algorithm for traffic classification into two classes i.e malicious and legitimate. The author has done Link Flooding attack in which the controller is disconnected from the data plane by sending slow legitimate traffic on the control channel. This type of attack was initially handled by the use of traffic filter but it was still a challenge. So the paper analyses the statistics during attack and then use Deep Learning algorithms to classify the traffic as malicious or legitimate. The author used mininet for real time traffic generation. It uses Artificial Neural Network for model training and then use the statistics collected as a test set for predicting the traffic class.

III. Distributed Denial of Service Attack and Countermeasures

In this section first we describe the Distributed Denial of Service(DDOS) attack which leads to data plane and control plane saturation. Later on, we present our proposed countermeasure to detect and mitigate the attack.

A. Distributed Denial of Service Attack

In Distributed Denial of Service Attack (DDOS) shown in figure 1, adversary aim is to saturate the target host with so many packets that it lets suffer the legitimate user in using the network services. Adversary performs the attack in such a way that it leads to both data and control plane saturation [8]. Adversary performs the following attacks:

- Flooding attack: Adversary performs the flooding attack to compromise the bandwidth and switch memory and thus controller delays access to benign users. The flooding attack is done by the use of the ping command from the source. For example if we assume h1 as a target host in our topology and h2, h3, h4 as the malicious hosts trying to attack H1. Syntax for one of the attack can be

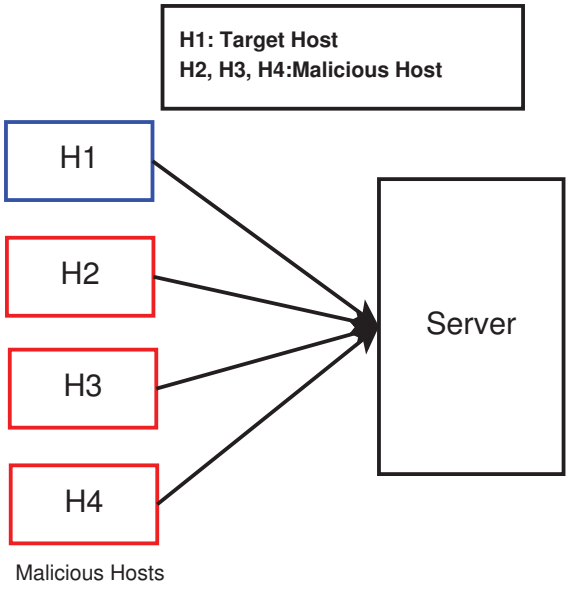

Fig. 1. DDOS Attack

given as: h2 ping –f h1 will send flood of packets from h2 to h1. Similarly, we can perform attack from h3 and h4.

- TCP-SYN Attack: Adversary performs the TCP Syn attack by directing the Syn Packet to the target but the destination host is yet to send the ACK to the source in the mean time adversary send other Syn Packet for connection establishment, so it become difficult for the target to handle the traffic.This is because of the reason as controller already allocates all its resources for the previous Syn requests and so controller reaches its saturation to handle the packets. To detect such an attack we have proposed a threshold-based approach which is explained in the next section.

B. Countermeasures

We proposed a countermeasure known as Continuous watch. It detects the attack by the malicious node by continuously analyzing the traffic statistics on switch. Detection and prevention of the attack [14] is done in two phases. In the first phase, attack is detected by analyzing the rate of change in the packet rate and bandwidth. In the second phase, the attack is prevented by changing the forwarding logic of the target host in the switch flow table.

1) Primary phase of detection: In this phase a thread is run continuously to monitor the switch. At regular intervals of time it collects the flow statistics which include various parameters which are mentioned in I & II

Packet count is used in the paper to calculate the packet rate. The packet rate value is used to check for the attack, whenever the packet rate is greater than a predefined threshold value of 100, the attack is detected and appropriate action is taken. Algorithm 1 shows the pseudo-code for attack detection.

TABLE I
FLOW STATISTICS

S.No	*Flow statistics*	*Meaning*
1	byte_count	Count of bytes
2	duration_n_secs	flow duration in seconds
3	priority	Priority of the flow
4	hard time out	set time when flow is removed
5	idle timeout	set idle time when flow is removed
6	len	lenghth of the message
7	match	can be IP,Mac address,Port No
8	packetcount	it is the total count of packets

TABLE II
PORT STATISTICS

S.No	Flow statistics	Meaning
1	txbytes	Bytes sent on a port in network
2	rxbytes	Bytes recieved on a port in network
3	txpackets	Packets sent on a port in network
4	rxpackets	Packets received on a port in network
5	txerrors	Errors during sending the packets
6	rxerrors	Errors during receiving of the packets
7	port id	Port number
8	datapath	switch id
9	duration	time during transmission
10	in_port	entry port
11	out_port	exit port

Algorithm 1 Algorithm for detection of attack in SDN

Input: Traffic statistics at switches
Output: Attack detected successfully

Initialisation :packetratelimit=100

For each flow collect the packet count and tx_ & rx_ byte statistics
for $i = packet1$ to $packet\_n$ **do**
  packetrate=packet_count/duration
  Bandwidth=((tx_bytes+rx_bytes)*8)/1024
  if (packetrate(i) $\geq packetratelimit$) **then**
    Attack detected
  end if
end for
return Flow with $SourceIP$,$destinationIP$

Secondly, we also compute the bandwidth for every 30 seconds. Bandwidth is computed for all the ports by specifying the OFPP_ANY [15] to the OFPFlowStatsRequest [5], [15] to measure the bandwidth of all the ports. We can specify the specific port number if we want to measure it from a particular port. Bandwidth is also found to be increasing high when the attack take place, but once the prevention technique is used then bandwidth will reduced. Flowchart in figure 2 shows the process of attack detection ad prevention.

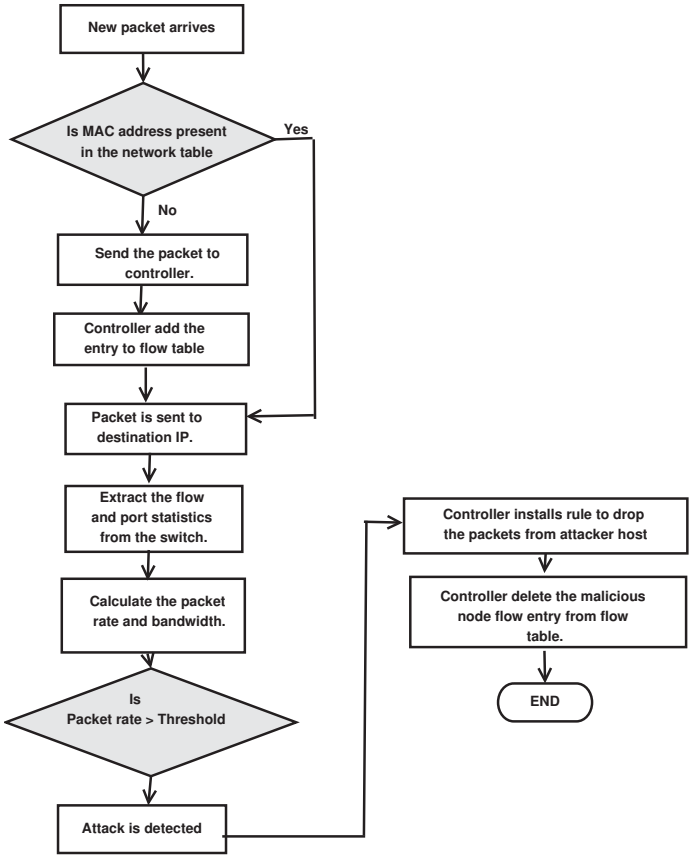

Fig. 2. Attack detection and prevention process

Algorithm 2 Algorithm for prevention of attack in SDN

Input: Flow for the Packet Rate,Bandwidth
Output: Deleted BlockedList

1: For the flow which has the identified packet rate and bandwidth
2: **for** $i = flow1$ to $flow\_n$ **do**
3: append the corresponding source IP of that flow into blocked list which is a list of IPs who have exceeded the treshold packet rate.
4: **if** (sourceIP(i) == $BlockedList$(i) **then**
5: Modify the action of that flow to drop the packets from source IP identified.
6: **end if**
7: **end for**
8: **return** Deleted $BlockedList$

2) Secondary phase & Countermeasures: In this phase, Attack prevention take place. The attack which is detected during the primary phase is prevented by changing the forwarding logic of the switches by modifying the flow rule in the flow table for discarding the further packets by using OFPFlowMod message [5] instead of forwarding. Algorithm 2 shows, the pseudo-code for the prevention of the attack.

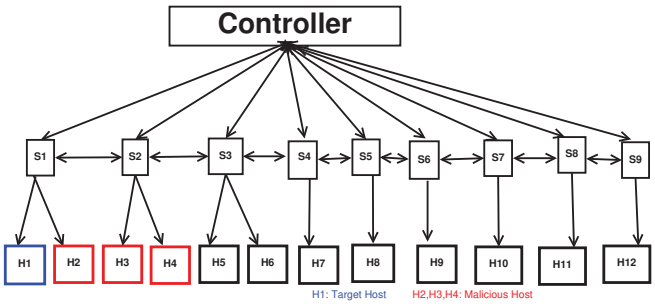

Fig. 3. Simulated topology

IV. Experimental Setup & Results

In this section, we show the setup conditions during experiments, hardware, and software used, their configurations and how we calculate the parameters associated with the experiment. Finally, the impact of attack with time is also shown.

A. Experimental setup

We used Mininet [5] as an emulator to analyze the network traffic, the attacks and their countermeasures along with Ryu as a network controller. The controller is written entirely in python and we have deployed our application logic as an application on Ryu.

Various simulation parameters are shown in Table III. Topology considered under investigation consists of 9 switches and 12 hosts is shown in figure 3. In the topology, H1 is the target host and Host 2, Host 3 and 4 are the malicious hosts.There are as many ports on a switch as the number of hosts. So, Host 1 and 2 are connected to switch S1. Host 3 and 4 are connected to switch S2 and switch S2 is connected to switch S1.

TABLE III
Simulation Environment

S.No	Parameters	Value
1	Host OS	Windows10
2	Guest OS	Ubuntu16.04
3	VirtualBox	5.1.26
4	Emulator	Mininet
5	Controller	Ryu
6	# Controller	1
7	# Switches	9
8	# Hosts	12
9	Protocol Used	OpenFlow
10	Graphical package	MiniEdit
11	Traffic Generation tool	Iperf, Curl, Ping
12	Controller Port Number	6653
13	Bandwidth	100 kbps
14	Simulation Time	300 seconds
15	Packet rate threshold used	100 packet per second
16	Statistics collection interval	30 seconds
17	Bandwidth plot interval	30 seconds

B. Performance Metrics

We have used two parameters for performance measurement to evaluating the effects of attacks and countermeasures.

- Packet Rate: It is the number of packets in the network per second. From the statistics collected total packet count can be collected from a switch but packet rate has to be calculated by subtracting the previous packet count from the current packet count and dividing by the total duration which will give the number of packets transmitted per second. Packet Rate is calculated as under :

$$packetrate = packetcount/duration \quad (1)$$

- Bandwidth: It is the total number of bytes transferred and received on a port. We are collecting the port statistics as mentioned in table II by calling OFPPortStatsRequest [15] which return the Port related statistics. Out of which we are using txbyte & rxbytes(transmitted bytes & received bytes) to calculate the bandwidth. By calculating the bandwidth we found that bandwidth suddenly shoots up in case of attack. Thus confirming that attack is done. We are calculating the bandwidth after every 30 seconds. Bandwidth is calculated as under:

$$bandwidth = (txbytes + rxbytes) \quad (2)$$

C. Result Analysis

To analyze the results we have plotted the graph of the two performance metrics without prevention technique and with prevention technique which will help us understand the scenario of attack in better way.

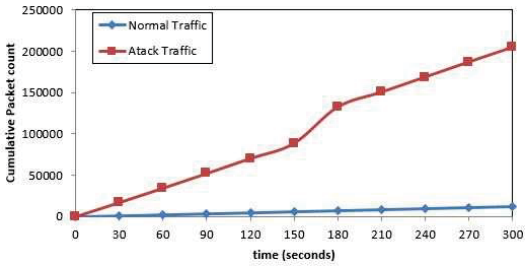

Fig. 4. Packet count vs time

Figure 4 shows the effect of attack on packet count. Figure 4 shows the effect of attack on number of packets. As can be seen from the figure that when there is no attack the number of packets over the network are in the range of (1238-12380) packets when measured after every 30 seconds. The packet count range increases in such a way because packet sending is done at uniform rate of 100 kbps by all the three hosts in the packet range from 1238 packets measured after every 30 seconds which equals to 12380 packets as the experiment is run for 300 seconds. Similarly, when a malicious node attack the target host (H1) the packet count shows a abrupt increase and reaches in the range of (18000-200000) packets. It is a strong indication of the attack. The increase in packet count is due to the malicious nodes(H2, H3, H4) attacked the target host (H1) at 100 packets per second by three malicious nodes respectively. Host 2 will send on port 2, so there will be 3000 packets every 30 seconds and on port 3 there will 6000 packets every 30 second. So three malicious nodes attack the target host and packet count increases in the interval of 9000 packets every 30 seconds and a total of 18000 as twice count for request and response.

For preventing the attack we have calculated the packet rate which is the per second count of the packets in the network. To check the attack, packet rate threshold value of '100' is used and if the packet rate at a particular switch crosses the threshold rate then the attack is detected and the control goes to the prevention module.

The prevention module alters the forwarding logic of the host from further forwarding the packets by changing the action part of the flow table to drop packets instead of forward and in this way the malicious source IP will not be able to send the packets again.

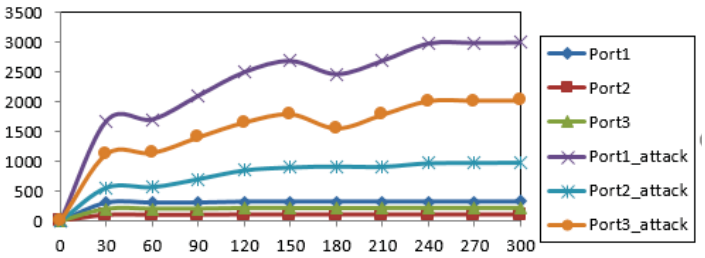

Fig. 5. Bandwidth during normal vs attack traffic.

Figure 5 depicts the bandwidth graph plotted between normal and attacked scenario. From the graph, we can infer that bandwidth of the network during normal traffic ranges in between (100-300 kbps) which shoots up during the attack in the range of (1600-3000 kbps) which is a indication of attack. Port1 in yellow is the port which is connected to host1 which is the target host and it is the sum of bandwidth at host 2, host 3 and host 4 who have attacked the target host at the rate of 100 packet per second.

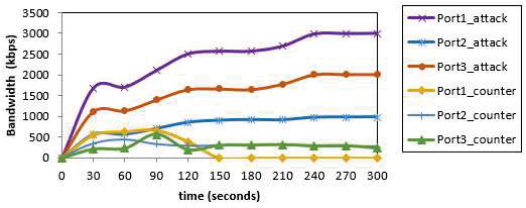

Fig. 6. Traffic Bandwidth in attacked vs after countermeasure applied

Figure 6 depicts the bandwidth of the network when countermeasure is applied which stays in the range of (400-600 kbps). It is because of the packet rate threshold which has been set to 100 packets per second, due to which when the attacker (H2, H3, H4) continue sending packets, they are dropped and port1 drop the packets and its bandwidth falls to zero.There is packet sending still done by Host 2,3,4 but Host 1 drop the packets can be seen from the figure 5 that port2, 3,4 have not fall to zero.

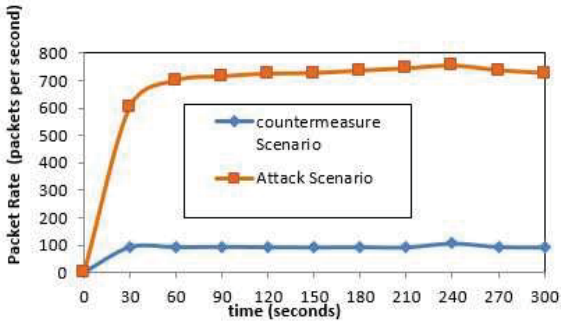

Fig. 7. Packet Rate during attack vs after prevention applied.

Figure 7 depicts the graph between packet rate before and after the countermeasure is applied. When the malicious hosts(H2, H3, H4) attack the target host (H1). The attack rate shoots up in the range of (600-744) but after applying the countermeasure whenever the packet rate goes above 100 the packets are dropped.

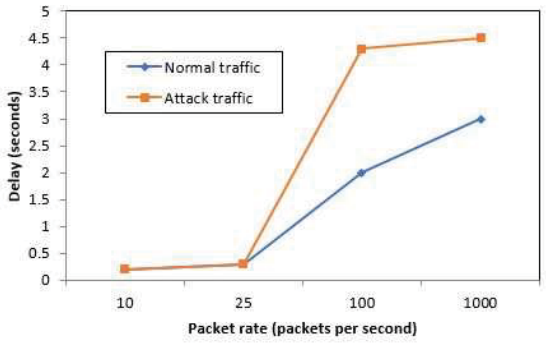

Fig. 8. Delay caused to legitimate user in normal traffic vs attack traffic.

Figure 8 shows the graph between delay caused to the legitimate user in sending the data when the attack take place. Graph depicts that when the attack rate increases from 100 pkt/sec, further increase is not increasing the delay. The delay decreases at higher attack rates because the controller is already busy in processing of previous packets. So delay reaches saturation.

V. Conclusion

In Software-defined networks, security [13] is a major concern and also data breaches is increasing with time. In this paper, we have implemented a method for detecting and preventing DDOS attack in SDN by collecting the flow statistics and port statistics at regular interval from the switches.If the calculated packet rate is above the defined threshold then the source is not allowed to send further packets by changing the forwarding logic in the flow table that maintained by switch.

In future, we will be implementing machine learning and deep learning approach for detection and prevention of such attacks using traffic logs.

References

[1] M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, Software-defined networking security: pros and cons, IEEE Communications Magazine 53 (6) (2015) 73–79. doi:10.1109/MCOM.2015.7120048.

[2] D. Kreutz, F. M. Ramos, P. Verissimo, Towards secure and dependable software-defined networks, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, ACM, New York, NY, USA, 2013, pp. 55–60. doi:10.1145/2491185.2491199.

[3] S. Shin, G. Gu, Attacking software-defined networks: A first feasibility study, in: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, HotSDN '13, ACM, New York, NY, USA, 2013, pp. 165–166. doi:10.1145/2491185.2491220.

[4] M. Brooks, B. Yang, A man-in-the-middle attack against OpenDayLight SDN controller, in: Proceedings of the 4th Annual ACM Conference on Research in Information Technology, RIIT '15, ACM, New York, NY, USA, 2015, pp. 45–49. doi:10.1145/2808062.2808073.

[5] www.mininet.org.

[6] Poisoning Network Visibility in Software-Defined Networks: New Attacks and Countermeasures. doi:10.14722/ndss.2015.23283.

[7] Hong, Kiwon, Youngjun Kim, Hyungoo Choi, and Jinwoo Park. "SDN-assisted slow HTTP DDoS attack defense method." IEEE Communications Letters 22, no. 4 (2017) 688-691.

[8] C. Yoon, S. Lee, H. Kang, T. Park, S. Shin, V. Yegneswaran, P. Porras,G. Gu, Flow wars: Systemizing the attack surface and defenses in software-defined networks, IEEE/ACM Transactions on Networking 25 (6) (2017) 3514–3530. doi:10.1109/TNET.2017.2748159.

[9] Kalkan, Kübra, Levent Altay, Gürkan Gür, and Fatih Alagöz. "JESS: Joint Entropy-Based DDoS Defense Scheme in SDN." IEEE Journal on Selected Areas in Communications 36, no. 10 (2018): 2358-2372.

[10] da Silva, Anderson Santos, Juliano Araujo Wickboldt, Lisandro Zambenedetti Granville, and Alberto Schaeffer-Filho. "ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN." In NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, pp. 27-35. IEEE, 2016.

[11] Meti, Nisharani, D. G. Narayan, and V. P. Baligar. "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks." In 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1366-1371. IEEE, 2017.

[12] Rasool, Raihan Ur, Usman Ashraf, Khandakar Ahmed, Hua Wang, Wajid Rafique, and Zahid Anwar. "Cyberpulse: A Machine Learning Based Link Flooding Attack Mitigation System for Software Defined Networks." IEEE Access 7 (2019): 34885-34899.

[13] K. K. Karmakar, V. Varadharajan, U. Tupakula, M. Hitchens, Policy based security architecture for software defined networks, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, SAC '16, ACM, New York, NY, USA, 2016, pp. 658–663. doi:10.1145/2851613.2851728.

[14] A. Shukhman, P. Polezhaev, Y. Ushakov, L. Legashev, V. Tarasov, N. Bakhareva, Development of network security tools for enterprise software-defined networks, in: Proceedings of the 8th International Conference on Security of Information and Networks, SIN '15, ACM, New York, NY, USA, 2015, pp. 224–228. doi:10.1145/2799979.2800009.

[15] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, O. Koufopavlou, Software-defined networking (SDN): Layers and architecture terminology, RFC 7426, RFC Editor, http://www.rfc-editor.org/rfc/rfc7426.txt (January 2015).