Detecting and Defending Denial of service Attack in Software defined Network Controller

Rahul parmani
BITS Pilani, Goa Campus
Goa, India
h20190021@goa.bits-pilani.ac.in

Mohit Varsha devarakonda
BITS Pilani, Goa Campus
Goa, India
h20190026@goa.bits-pilani.ac.in

Srivijay Ram Prathap Nandikolla
BITS Pilani, Goa Campus
Goa, India
h20190559@goa.bits-pilani.ac.in

ACM Reference Format:
Rahul parmani, Mohit Varsha devarakonda, and Srivijay Ram Prathap Nandikolla. 2020. Detecting and Defending Denial of service Attack in Software defined Network Controller. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Software defined network is basically separation of control plane and data plane which is also called as the forwarding plane, which consists of the traditional networking elements such as routers and switches. The control plane contains a centralized controller which has the main control of how the data traffic is will be managed and how it will be handled. On the other hand the data plane which is also called as the forwarding plane manages the forwarding or dropping of the data traffic.

Software-Defined Networking (SDN) Architecture broadly consists of three layers: Application layer, Control layer and Infrastructure layer.

The infrastructure layer consists of different forwarding devices such as switches and routers. The control layer is the layer where the controller resides which is responsible for all the decisions to be taken for forwarding. Control layer lies in middle and it exposes two types of interfaces – Northbound and Southbound. Northbound interface is meant for communication with the upper, Application layer and Southbound interface is meant for communication with lower, Infrastructure layer of network elements.

In Application layer different types of applications can be developed by using all the network information about network topology. Southbound communication from control layer to infrastructure layer (network switches) is done through Openflow protocol. Software-Defined Networking (SDN) Architecture broadly consists of three layers: Application layer, Control layer and Infrastructure layer. The infrastructure layer consists of different forwarding devices such as switches and routers. The control layer is the layer where the controller resides which is responsible for all the decisions to be taken for forwarding. Control layer lies in middle and it exposes two types of interfaces – Northbound and Southbound. Northbound interface is meant for communication with the upper, Application layer and Southbound interface is meant for communication with lower, Infrastructure layer of network elements. In Application layer different types of applications can be developed by using all the network information about network topology. Southbound communication from control layer to infrastructure layer (network switches) is done through Openflow protocol.

In Software defined networks as the control plane and data plane are completely decoupled all the routing information, security measures, network management will reside in the control plane.So protecting controller is a very important task in a SDN in a software defined network.

Denial of service attacks are kind of cyber-attack in which the attacker intend to stop the services from a particular server or computer to intended users. Denial of service attack is achieved by flooding the targeted machine or resource with surplus requests in an attempt to overload systems and prevent some or all legitimate requests from being fulfilled.

2 RELATED WORKS

[1]A. F. M. Piedrahita et al. proposed a light weight method for Detecting and Mitigating a Denial of Service attack in Software Defined Networks. The method works even in the case of traffic being not differentiable between legit and malicious. The algorithm uses computation of bandwidth across various flows using a background daemon process on the switchers or routers. Bandwidth is incremented for flows that are well behaved i.e. their current usage doesn't exceed the mean Bandwidth, in the other case the Bandwidth is decremented proportional to the over usage. Since the algorithm cannot distinguish between malicious and legitimate flows, the same penalty is applied even if the flow is legitimate.The above proposed mechanism is lightweight and computationally efficient as it doesn't require heavy resources and it doesn't have to inspect packets for headers and other fields.

[2]J. S. Maddu et al. proposed the approach in order to limit the number of packets that are sent to the controller in case of mismatch in the flow table entry, caused as a result of Denial of Service attack. Simultaneously reducing the loss incurred to legitimate users if they fall in the same flow of attackers.The proposed system works in detection and defence stages, with the help of two additional modules named packet migration module and data cache module. Once an attack is detected with the help of global flood count and threshold, the defence mechanism then divides the flow partially to the data cache where it is processed later and the other to the controller. The algorithm mitigates the Denial of service attack but doesn't completely eliminate it. The algorithm although

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

effective, requires the addition of two extra modules to the existing architecture. The size of the data plane cache to be used also has to be addressed.

[3] R. Nagarathna et al. devised a solution to the problem of Host location hijacking which leads to the Denial of Service attack, in the scenario when the attacker impersonates legitimate users inside the data plane. The approach uses Iterative Dichotomiser 3 a machine learning based classifier to classify host as malicious or benign based on various attributes which include CPU utilization, bandwidth consumption etc. Once a packet-in message is received is compared with Host information, in case of exact match it is processed as usual, in the other case it's sent to the classifier. The algorithm is light weight and effective and functions even if the host location is unknown. A timer based blocking system provides falsely classified users another chance.

[4]H.Rathore et al. Has proposed a human immune system based solution to tackle with the problem of Denial of Service attack in Software Defined Networks. The proposed system uses mathematical differential equations and Runge-Kutta variable methods in order to control the bandwidth at a switch corresponding to the change in the rate of flow.Proper initialization of constants in the differential equations are heuristic.The model is not computationally heavy and does not require huge memory, hence can be stored in Ternary Content Addressable Memory TCAM of the switches.

[5] L. Zhang et.al. proposed a Port Hopping DOS Mitigation (PH-DM) which is a typical kind of Moving Target Defense Technology which dynamically allocates the ports under service to unused ports. PH-DM Technique confuses the attackers as they are unable to locate the opened ports which is used by the network services which increases the time and cost of the attack.The trusted/ authorized clients will have the knowledge of the opened ports which will be dynamically changing time to time. When the attacker tries to make a connection if the destination ports matches the open ports during the whole transmission then the transmission is made. While in between, if the destination port does not match the open port then new flow entries will be made to drop the packets.PH-DM mechanism has a very high requirement of synchronization between the communication sides. The results showed an acceptable overhead processing range of (2.1%-4.9%) on the SDN controller which is not heavy.

[6] T. Chin et.al. proposed an approach that focuses on detecting a type of DOS attack called as TCP-SYN Flood attack in which while performing a three way handshake the client uses a fake spoof IP address and keeps on sending SYN packets to the server to fill up the buffer space of the server so that it can't take other request and may eventually crash.This approach proposed a collaborative detection and containment mechanism which makes use of a Monitor which is distributed over the network and a correlator which resides on the controller side.The monitor consists of rules and signatures which detects anomalies in the network traffic. When the packets are sent to the server, the packets are mirrored and sent to the monitor for inspection. If the monitor detects anomalies in the network traffic it raises an alert to the correlator which in turn gathers more information from the switch for the inspection of packets and if an attack is detected, the controller makes new flow entries into the switch to block the packets.This approach proposed an efficient solution which is able to handle large amount of traffic but is time consuming in which the monitor takes time to analyze the packets and raise alerts, the correlator takes time to get flow entries from switch and act accordingly. Further, more improvement can be made by increasing the number of monitors or use more powerful machines.

[7] P. M. Ombase et.al proposed an approach that aims to detect and mitigate the Denial of Service attack in the Software defined Network (SDN) using Intrusion Detection Systems(IDS). The IDS used in their approach is SNORT which is a signature/rule based detection approach and BRO anomaly based detection approach.Bro/Snort IDS is setup between the Controller and the Switches and thus the traffic from the switch to the Controller pass through them first and filtering and matching of packets will be done. Snort IDS analyzes the packets and uses predefined activity patterns to identify some known attack in the networks and matches the activity to these patterns which is also known as rules or signatures.If signature is matched with the current ongoing activity it takes appropriate actions to mitigate it such as alert or drop. Bro IDS performs analyses of the network and makes a log entry of the normal traffic and studies the network. Whenever a suspicious activity is detected it takes necessary actions upon it.This approach does solve detecting attacks in the network but it increases the computation as the packets will first go through the IDS. This method also suffers with false alarms which can be a topic of research to reduce it in the future.

[8] J. Galeano-Brajones et.al proposed an entropy based approach to detect and mitigate Denial of Service attack in Software Defined Network (SDN). Shannon entropy is used to find changes in the normal distribution of network traffic to identify anomalies. For the analysis of the traffic an initial learning phase is done to feed the algorithm about the normal flow of traffic for calculating the average and typical deviation which are the upper and lower limits of the entropy. For detecting the attacks, the process is repeated under regular intervals and entropy is analyzed to detect if it is over or under the limits which is defined in the network, if it is then an attack is detected. Once the attack is detected, by noticing significant changes in the traffic features, the counter measure is taken by the controller by adding a new flow entry in the flow table to stop the attack with an instruction to drop these packets. In the future, analysis of different statistical-based metrics can be done, which can be used for detection of DOS attack.

[9] S. Wang et.al. proposed using of monitor mechanisms. SDN controller have the complete information of the network. The information like packet-In messages, Switch ID from where these packets are coming from etc., can be helpful to detect a Dos Attack is SDN. To detect a Dos Attack, Controller capability is used to make certain thresholds, the count of Packet-In messages along with the Switch-Id this algorithm updates the thresholds at different levels of the network at particular time periods. On detecting Dos depending upon the type of the attack either particular switch or the attacker can be blocked. This algorithm improves its success rate of detecting the attack with increase of attacks frequency, but at very high frequency rates the attacks is still able to break through the SECO. The Network is still vulnerable to attack. Using of monitor mechanisms is a good standard approach to detect Dos. Updating the thresholds periodically opens a window for the attacker. Detecting the attack in its early stages will be challenging.

[10] G. Shang et.al. proposed a Flood Defender mechanism. SDN Controller gives freedom to improve its architecture. Without using any additional hardware to defend against the Dos attack the idea is to improve its architecture and dividing the detection and defending against Dos attack into different modules and distributing traffic.Implement a Flood Defender mechanism in the controller such that with modules like Flood detection, Attack detection, Table-entry miss engineering, Packet Filtering. To avoid the traffic between the attacked switch and the controller the traffic will be distributed to the detoured switches and then these packet-in messages from each switch will be filtered.As they are different modules present in this most of the network traffic will be occupied by them. When a network with all of its switches are busy and distributing traffic to them might not only effect the current attacked switch but also can affect other switches also. Usually in a network is mostly likely to have at least few switches with less traffic. Even with single neighbour switch this model improves CPU utilization and No additional resources also not required. Not recommended to network which have continues traffic.

[11] M. Zhang et.al. proposed to filter the packets in the data plane instead of implementing the mechanism in the control plane. This filtering is done by dividing the legitimate packets from the attacker's packets by the features of packets. There are two modules in this mechanism. First, Using Source address validation to find the packet came from a trustworthy host or not. This can be done by dividing the ports into different categories and then making validation only on not trustworthy port packets. Second, Using Stateful Packet Supervision the packets which have a legitimate source address but still forged by the attacker are identified and filtered. This mechanism is not tested on a real world network. The results when deployed on a testing environment shows minimum CPU utilization and good efficiency. The implementation of such a mechanism is very hard and complex. The results shows that it is more efficient than standard flood defensive mechanisms like floodshield, floodgaurd. Deploying it into a real world application requires to take care about lot of other issues like what is the switch itself is tampered.

[12] T. Abhiroop et.al. proposed different machine learning models to detect exhaustion of TCAM, which is the primary aim of this. They extracted features like Average number of Packets per Flow, Average number of Bytes per Flow, Average Timeout per Flow-rule, Percentage of Pair Flows, Rule FlowMod Disparity to train and classify the state of controller whether TCAM is in normal condition or being attacked. To train models like Support Vector Machine, Neural Network, and Navies Bayes model are used.Out of three models Neural networks and Naive Bayes approach showed 100% accuracy on test data set. As the model is trained by simulated data, not on real world data the accuracy of the model when deployed on real world applications is unpredictable. Machine learning models gets better with more appropriate data so these approach keeps on improving with more and more use of SDN. Deployment and computation of trained model is very effective in cost of time and computation.

3 NETWORK MODEL

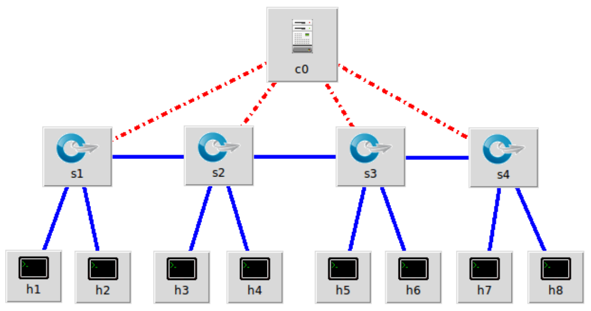


Figure 1: The Network Model with one controller,four switches and eight hosts.

4 PROBLEM STATEMENT

Denial of service attack can be implemented in two ways, In First scenario is to alter the switch in such a way that instead of receiving any new packets the switch itself generates packet-in messages and sends it to the controller in such a way that the controller will be busy with handling this packet-in messages.

In second scenario, the attacker keep on sending the packets with a very high rate. These packets will be generated in such a way that, they doesn't match with any of the previous flow rule entries. So the switch sends a packet_in message to the controller. Controller sends a packet_out message and flow rules to the Switch. As the incoming packet rate is very high the controller keeps on sending these packet_out message to the malicious packets. This wastes the CPU utilization of the controller, resources of the controller, bandwidth of the channel, TCAM (Ternary Content Addressable-Memory) space, Unable to serve the legitimate requests.

If the DoS attack is successfully carried out on the controller (using scenario one or two), the normal requests to set up new connections to the controller will be denied because there are not enough resources in the controller for handling those packets.

A counter at every switch which will examine the incoming packet_in messages and update the counter. A threshold will be set at every switch. If the counter value in a certain period of time exceeds the threshold value, that attacker host will be blocked for a fixed amount of time. Thus, the DOS attack will be countered and the other host's requests can be severed. After the fixed time is over, the blocked attacker host will be unblocked and if it still floods with packets then it will be blocked again.

Considering only the packet_in messages might be misleading, so in our proposed solution we consider the ratio of packet_in messages to the transmitted messages. We will try to detect the Denial of Service attack, much earlier using the above ratio.

5 PROPOSED SOLUTION

This algorithm uses different counters to detect the DOS attack. The following variables should be considered for better understanding,

- i=(1,2,3,...n) represents the controller port.
- j=(1,2,3,...m) represents the switch port.
- γ_{ij} is number of packet_in messages from switch i and host j.

$$\gamma_{ij} = \sum Packet\_in_{ij}$$

- β_i is number of packet_in messages received from a switch i.

$$\beta_i = \sum_{j=1}^{m} \gamma_{ij}$$

- β is the sum of all packet_in messages received in from all the switches.

$$\beta = \sum_{i=1}^{n} \beta_i$$

- ω_i is calculated as the number of Packet_In messages received to make the controller CPU utilization reach σ% (Example 80%).

$$\omega_i = \sum_{i=1}^{n} \beta_i, \text{ for } \sigma\%$$

- α_i is counter threshold value for switch i.
- λ_i is the number of packet_in messages for switch i.
- τ_i is counter threshold value for switch i calculated for every T interval of time.

$$\tau_i = \omega * \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

- ρ_{ij} is 1 - the ratio between the number of packer_in messages received to the total number of transmitted messages from the switch i and host j.

$$\rho_i = 1 - \frac{\beta_i}{\theta_i}, \text{where } \theta_i \text{ is the number of total transmitted messages from switch i.}$$

For every T interval of time the τ_i will be updated with the new value calculated. By default $\alpha_i = \frac{\omega}{n}$.When a new packet_in message is received to the controller all the counter values will be updated according to the packet source and destination information.

The algorithm works as follows, There are two functions. One is when the threshold α_i is exceeded by the β_i but β is less than ω then as the overall packet_in message count is not effecting the CPU utilization the thresholds α_i of the switch will be updated by the value of τ_i.

$$\text{if } \beta_i >= \alpha_i \text{ and } \beta <= \omega \text{ then } \alpha_i = \tau_i$$

Second is when threshold α_i is exceeded by the β_i and β is greater than ω then as the CPU utilization is effected by the number of Packet_in messages receiving it is considered as a DOS attack.

$$\text{if } \beta_i >= \alpha_i \text{ and } \beta > \omega \text{ then DOS Detected.}$$

Algorithm 1 Detecting DOS

```
1: procedure DOS_DETECTION($\omega, \beta, \beta_i, \alpha_i$):
2:     if $\beta_i >= \alpha_i$ and $\beta <= \omega$ then
           $\alpha_i = \tau_i$
3:     if $\beta_i >= \alpha_i$ and $\beta > \omega$ then
           DOS Detected
```

To detect the host which is the attacker we can use γ_{ij} and the host which has the maximum γ_{ij} is the attacker. Using this switch port and controller port i,j we can drop the packets from that host for a particular period of time. Still the controller can handle requests form all other hosts.

The value ω is computed in lab depending upon the hardware specifications of the controller.The CPU utilization σ can be decided dynamically depending upon the ρ_{ij}.For this σ is bounded by two values, one is the minimum_σ and another is maximum_σ. This ρ_{ij} can be used as a factor to decide the reliability of the host, When an attacker intended to implement a DOS attack he generates packets such a way that the number of packet_in messages generated by those packets is as high as possible. So ρ_{ij} gives a value between 0 and 1 which can be a score to identify the host intention of implementing a DOS attack.

When the value of ρ_{ij} is near to 0 it shows that most of the transmitted packets from that host is creating a packet_in messages. In this situation we decrease the σ towards the minimum_σ which imply the algorithm to decrease the counter threshold values and ω. As the threshold is decreased the DOS attack is detected in less time.

When the value of ρ_{ij} is towards 1 it shows that most of the transmitted packets from the host not generated the packet_in messages so it is believed as a more reliable host so we increase σ towards maximum_σ which imply the algorithm to increase the threshold counter values and ω. This gives more time to the host before deciding it as a DOS attack.

Algorithm 2 Algorithm to find the σ using ρ_{ij}

```
   procedure SIGMA($\gamma_{ij}, \theta_{ij}$)
2:     $\rho_{ij} \leftarrow \frac{\gamma_{ij}}{\theta_{ij}}$
       $min\_\sigma \leftarrow 60\%$
4:     $max\_\sigma \leftarrow 90\%$
       $\sigma \leftarrow min\_\sigma + ((max\_\sigma - min\_\sigma) * \rho_{ij})$
6:     return $\sigma$
```

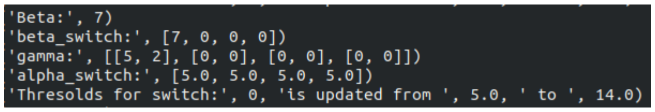
This σ is used to change ω dynamically.

6 SIMULATION AND RESULTS

The simulation environment used is mininet to create the Network. POX is used as the controller for the network.The Network consists of 4 Switches and 8 Hosts, Two hosts for each switch as stated in the network model[3]. The hosts are named as h1,h2,h3,h4,h5,h6,h7 and h8. All the counters defined in the proposed solution are defined in POX controller. When a packet_in message comes to the controller it will be analysed by the controller to update all the counter according to the algorithm.

To produce the packet_in messages ping between the hosts can be used as the host generates packet_in message and send them to the controller.The timer value to update the β value is set to 20 seconds. The ω value is initialized with value of 20.All other counters are initialized to 0 and they will be updated according to the packet_in messages received by the controller.The initial Thresholds will be $\frac{\omega}{n}$ for all the switches which is 20/4(5). So α_i=[5,5,5,5].

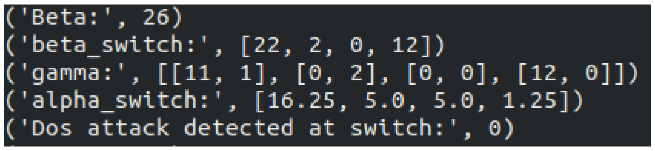
To make it more convenient to observe the results we are printing the counter values in the POX terminal.When h1 ping h2 is started and after the number of packets_in messages from h1 exceeds the default threshold of α_i=(5) but the β=(7) still less than the ω=(20) this comes under the condition as NOT a DOS attack but we have to update the α_i=(5) with the τ_i=(14). This is because at time point of time only messages from h1 are generated so the Switch 1 is the most used switch till now. So the value of threshold of that switch should be more than other switches. The updated Thresholds are [14,5,5,5]



```
'Beta:', 7)
'beta_switch:', [7, 0, 0, 0])
'gamma:', [[5, 2], [0, 0], [0, 0], [0, 0]])
'alpha_switch:', [5.0, 5.0, 5.0, 5.0])
'Thresolds for switch:', 0, 'is updated from ', 5.0, ' to ', 14.0)
```

Figure 2: The counter values when thresholds are updated

To generate a DOS attack this ping can be continued such that the packet_in messages from switch 1 should be greater than the threshold of switch 1 and the β should be greater than ω When this happened it is detected as DOS and the packet_in messages from this switch should be dropped further.



```
('Beta:', 26)
('beta_switch:', [22, 2, 0, 12])
('gamma:', [[11, 1], [0, 2], [0, 0], [12, 0]])
('alpha_switch:', [16.25, 5.0, 5.0, 1.25])
('Dos attack detected at switch:', 0)
```

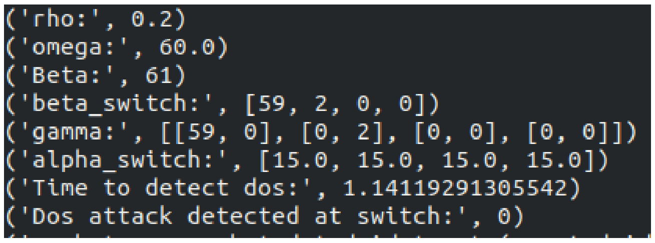
Figure 3: The counter values after the DOS is detected

To simulate a DOS attack we used hping3 software. This generates thousands of packet_in messages in seconds.We simulated a DOS attack from host1,When started this DOS attack without any defence mechanism the CPU utilization went 100% and complete network went down.No other host is able to ping any other host.

After applying this algorithm, the mechanism successfully defended the network from DOS attack and dropped all packets from host1.Complete network is still working and every host is able to ping each other except host1. This shows the mechanism works successfully.

To test the working of ρ_{ij} to dynamically update ω started a DOS attack from host1 with min_ω=50 and max_ω=100 so that when ρ_{ij} is varies between 0 to 1 the ω will be varied from 50 to 100.

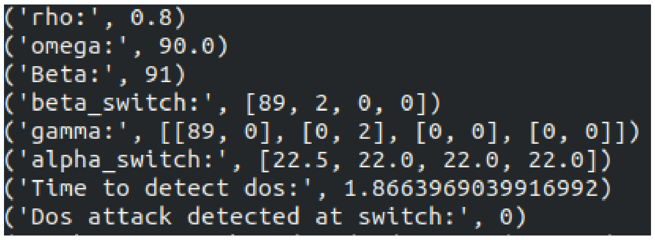
When tested while the ρ_{ij} is 0.2 which means it says that the host is intended to attack the network the ω is changed to 60.This detected the DOS attack in 1.14s time and the it took only 60 packets to detect the DOS attack.



```
('rho:', 0.2)
('omega:', 60.0)
('Beta:', 61)
('beta_switch:', [59, 2, 0, 0])
('gamma:', [[59, 0], [0, 2], [0, 0], [0, 0]])
('alpha_switch:', [15.0, 15.0, 15.0, 15.0])
('Time to detect dos:', 1.14119291305542)
('Dos attack detected at switch:', 0)
```

Figure 4: The time and ω to detect the DOS attack with ρ_{ij} = 0.2

The same algorithm and same DOS attack is simulated again but with a ρ_{ij}=0.8 which means the host is not intended to simulate an attack. The ω value is changed accordingly to 90 and gave more time to the host before detecting it as DOS attack. The time took to detect the DOS attack is 1.86s and after receiving 90 packet_in messages.



```
('rho:', 0.8)
('omega:', 90.0)
('Beta:', 91)
('beta_switch:', [89, 2, 0, 0])
('gamma:', [[89, 0], [0, 2], [0, 0], [0, 0]])
('alpha_switch:', [22.5, 22.0, 22.0, 22.0])
('Time to detect dos:', 1.8663969039916992)
('Dos attack detected at switch:', 0)
```

Figure 5: The time and ω to detect the DOS attack with ρ_{ij} = 0.8

7 CONCLUSION

The algorithm has successfully detected and defended the Network from DOS attack. As we can see from the results the counter threshold values are updated dynamically according to the network traffic in that switches. This gives more flexibility for the network to use this algorithm in any situation without effecting the dynamic property of traffic.

As the algorithm also uses the ratio between the packet_in messages to the total transmitted packets this help it to idealize the nature of host either as a attacker or as a legitimate user. It gives more CPU utilization and resources to a legitimate user and detects the attacker much faster than a normal counter based mechanism.

REFERENCES

[1] A. F. M. Piedrahita, S. Rueda, D. M. F. Mattos and O. C. M. B. Duarte, "Flowfence: a denial of service defense system for software defined networking," 2015 Global Information Infrastructure and Networking Symposium (GIIS), Guadalajara, 2015, pp. 1-6.

[2] J. S. Maddu, S. Tripathy and S. K. Nayak, "SDNGuard: An Extension in Software Defined Network to Defend DoS Attack," 2019 IEEE Region 10 Symposium (TENSYMP), Kolkata, India, 2019, pp. 44-49.

[3] R. Nagarathna and S. M. Shalinie, "SLAMHHA: A supervised learning approach to mitigate host location hijacking attack on SDN controllers," 2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN), Chennai, 2017, pp. 1-7

[4] H. Rathore, A. Samant and M. Guizani, "A Bio-Inspired Framework to Mitigate DoS Attacks in Software Defined Networking," 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), CANARY ISLANDS, Spain, 2019, pp. 1-5.

[5] L. Zhang, Y. Guo, H. Yuwen and Y. Wang, "A Port Hopping Based DoS Mitigation Scheme in SDN Network," 2016 12th International Conference on Computational Intelligence and Security (CIS), Wuxi, 2016, pp. 314-317.

[6] T. Chin, X. Mountrouidou, X. Li and K. Xiong, "Selective Packet Inspection to Detect DoS Flooding Using Software Defined Networking (SDN)," 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops, Columbus, OH, 2015, pp. 95-99.

[7] P. M. Ombase, N. P. Kulkarni, S. T. Bagade and A. V. Mhaisgawali, "DoS attack mitigation using rule based and anomaly based techniques in software defined networking," 2017 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, 2017, pp. 469-475.

[8] J. Galeano-Brajones, D. Cortés-Polo, J. F. Valenzuela-Valdés, A. M. Mora and J. Carmona-Murillo, "Detection and mitigation of DoS attacks in SDN. An experimental approach," 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS), Granada, Spain, 2019, pp. 575-580.

[9] S. Wang, K. G. Chavez and S. Kandeepan, "SECO: SDN SEcure COntroller algorithm for detecting and defending denial of service attacks," 2017 5th International

Conference on Information and Communication Technology (ICoIC7), Malacca City, 2017, pp. 1-6.

[10] G. Shang, P. Zhe, X. Bin, H. Aiqun and R. Kui, "FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, Atlanta, GA, 2017, pp. 1-9.

[11] M. Zhang, J. Bi, J. Bai and G. Li, "FloodShield: Securing the SDN Infrastructure against Denial-of-Service Attacks," 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, 2018, pp. 687-698.

[12] T. Abhiroop, S. Babu and B. S. Manoj, "A Machine Learning Approach for Detecting DoS Attacks in SDN Switches," 2018 Twenty Fourth National Conference on Communications (NCC), Hyderabad, 2018, pp. 1-6