International Journal on Emerging Technologies **11**(2): 627-632(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

A Novel Hybrid Mitigation Technique against DoS Attacks in Software defined Network with Entropy, SVM and Reinforcement Learning

***Trupti Lotlikar*[1] *and Deven Shah*[2]**
[1]*Assistant Professor, Department of Information Technology,*
Terna Engineering College/Fr.CRIT Navi Mumbai (Maharashtra), India.
[2]*Associate Professor, Department of Information Technology,*
Thakur College of Engineering Kandivli, Mumbai (Maharashtra), India.

(Corresponding author Trupti Lotlikar)
(Received 23 December 2019, Revised 25 February 2020, Accepted 27 February 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Software Defined Network (SDN), also known as a Smart Network, as it performs significant role in regulating and managing number of heterogeneous networks. Tragically, SDN faces a great deal of security issues that may seriously influence the system activities if not appropriately tended to. In the SDN network, there is centralization of the controller and hence any DoS attack will cause the entire system to collapse. The DoS attack on centralized model brings about huge challenge of communication overhead, packet delay and loss of genuine packets. Another challenge is there is a lack of research on finding a common methodology for intelligently evaluating the security of SDN controllers. Thus, the paper contributes by evaluating an intelligent hybrid method for detection and mitigation of DoS attack with Entropy, SVM and Reinforcement Learning with Markovian Process model. The paper proposes a profound reinforcement learning based system, which can intelligently gain proficiency in learning the optimal mitigation policies under various attack scenarios and mitigate the DoS flooding attack in real time. Practical experiments are conducted in the Mininet environment, to defend against a wide range of DoS flooding attacks such as TCP SYN, UDP, and ICMP flooding and proves that the proposed novel hybrid mechanism can be an effective against DoS attacks, causing benign traffic to keep flowing, keeping the network working. The framework also proposes a novel flow based algorithm which can determine the attacker in crucial attack.

Keywords: Software defined network; Denial of Service, Security, Support Vector Machine, Reinforcement Learning, Markovian Model.

Abbreviations: SDN, Software Defined Network; DoS, Denial of Service; SVM, Support Vector Machine; MDP, Markovian Decision Process.

I. INTRODUCTION

Software Defined Networking (SDN) is a favorable solution for tending to difficulties of future networks [1]. SDN in contrast to traditional networks is a network technology where the control plane logic is decoupled from the forwarding plane and has the ability to control, change and manage network behavior dynamically through software via open Application Program interface (API) [3]. This causes the control plane to supervise and control network performance by means of programming control. In spite of SDN's favorable characteristics, for example, adaptability, transparency and reasonable cost, it has a few disadvantages that are to a great extent incited by the centralized control view. Security is one of the most important threat identified with centralization. Therefore, Denial of Service (DoS) attacks suggest critical security issue in SDN [2].

In traditional networks, both the planes are combined on the same devices, allowing each device to make its own forwarding decisions based on some distributed routing protocols [11]. On the contrary, SDN allows for the control-plane to have a global and centralized view of the network. Some of the Research Gaps identified were, the earlier detection methods proposed could either detect or mitigate the Dos Attacks, so there is no framework which both detects and mitigates the attack. Secondly, prior mitigation techniques proposed included human intervention either through supervised learning or using training data. Hence the research intends to provide a solution to mitigate the above-mentioned situations with an intelligent, self- learning Framework. Reinforcement Learning (RL) is thus implemented as a novel computational approach for understanding and automating goal-directed learning and decision-making.

Apart from this a novel flow-based algorithm is proposed which reduces our search of attacker machine to a limited number of hosts associated with the switch identified, reducing the time for mitigation.

In this paper, to detect DoS Attacks within SDN, we propose a hybrid method of Entropy, a machine learning algorithm e.g. Support Vector Machine (SVM) for classifying network traffic as normal or anomalous and Snort IDS for deep packet inspection is used. Mitigation of a DoS attack is proposed using novel, intelligent deep Reinforcement learning based approach which works on Markovian Process mathematical model and a novel algorithm is proposed to find the attacker in the mitigation phase.

The remaining paper is classified as follows. Section II discusses the proposed methodology of attack detection using Support Vector Machine (SVM) and mitigation using Reinforcement Learning. A novel algorithm is

proposed in mitigation module that points to the attacker with flow concept. Section III discusses the results and Section IV gives the conclusion.

II. PROPOSED METHODOLOGY

A. Detection of Dos Attack in SDN with Entropy and Support Vector Method (SVM)

The technique used to detect DoS, described in this section is Support Vector Machine (SVM), which is a supervised machine learning technique. It includes segregating and focusing points in p-dimensional space, with a hyperplane that is a level relative subspace of p 1 dimension [17]. For example, in two dimension, a hyperplane is a level one-dimensional subspace - as it were, a line. In the proposed method, SVM is used to classify, based on several feature vectors [18], whether a given scenario of flow packets during an experimental time slot is a DOS attack or not. In Entropy, we find the entropy values of the features of source, destination IP's and port's and then using these features we prepare a training data set which can be used to train the SVM [16]. The SVM will further classify whether the traffic is normal or attack traffic. Hence we say its an hybrid methodology using entropy, SVM and reinforcement learning. Once we are sure about the attack, we proceed further with Reinforcement learning techniques to mitigate the attack [15].

In the SVM literary work it is entirely expected to utilize +1 and -1 to signify the two classes. For a hyperplane characterized by weight w and inclination b, a direct discriminant is given by

$$w^T x+b=g>0 \quad \text{class+1}$$
$$<0 \text{class}-1 \qquad (1)$$

The numerical interpretation of a hyperplane is very straightforward. In two dimension, a hyperplane is characterized by the condition,

$$\beta_0 + \beta_1 X_1+\beta_2 X_2=0 \qquad (2)$$

for parameters β_0, β_1 and β_2.

We state that, the above condition, characterizes the hyperplane. That is, any $X = (X_1, X_2)^T$ for which the above condition holds is a point on the hyperplane. It can be handily applied to the p-dimensional setting:

$$\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_p X_p=0 \qquad (3)$$

This condition characterizes a p-dimensional hyperplane. Again as in if a point $X = (X_1, X_2, \ldots, X_p)^T$ in p-dimensional space (for example a vector of length p) fulfills the above condition, at that point X lies on the hyperplane [3].

Presently, assume that X doesn't fulfill the given condition; rather,

$$\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_p X_p>0 \qquad (4)$$

At this point this reveals to us that X lies one side of the hyperplane. Then again, if

$$\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_p X_p<0 \qquad (5)$$

at that point X lies on the opposite side of the hyperplane. So we can think about the hyperplane as partitioning p-dimensional space into equal parts. One can without much of a stretch decide on which side of the hyperplane a point lies by essentially ascertaining the sign indication of the left hand side of (2)

On the off chance that f(xj) is positive, at that point we allot the test perception to class 1. If f(xj) is negative, then we assign the test observation to class-1 [20].

The graph below shows the detection of the attack with all the points below the the hyperplane indicate an attack. 3 features are considered i.e the src ip, destination port and number of bytes [13]. The normal traffic and attack traffic is depicted in the Fig. 1 below. Similarly the Fig. 2 and 3 give the snapshots of features mapped with packets per second(pps) and bytes per second(bps) respectively.

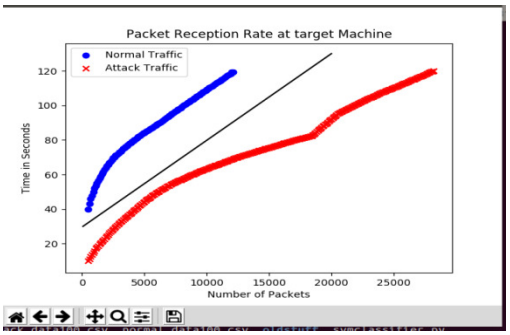

Fig. 1. Snapshot of SVM output.

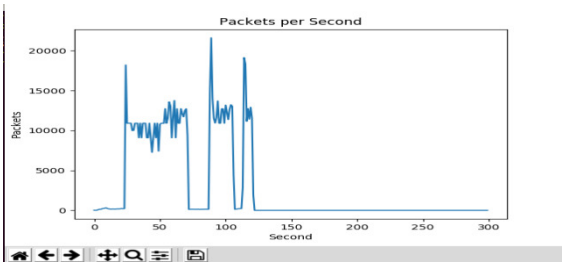

Fig. 2. Snapshot of feature Output packets per second.

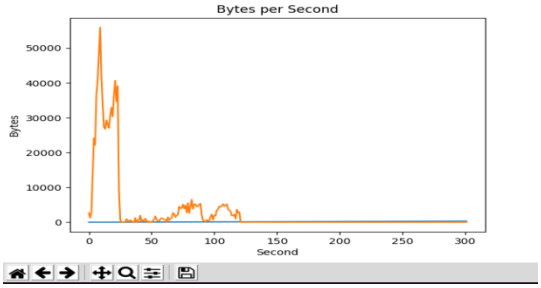

Fig. 3. Snapshot of feature Output Bytes per second.

The experimentation could easily proof the detection of the DoS attack. Training of data was done with machine learning. All the data values above (hyperplane) 1 value was taken as no attack and all the values below the (0) was taken as attack. Receiver Operating Characteristic (ROC) curve is a useful tool when foreseeing the likelihood of a binary result. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for various distinctive threshold values [9] somewhere in the range of 0.0 and 1.0, predicting the probability of a binary outcome. Fig. 4 depicts the RoC Graph of Training Data. As from the graph it is clear that SVM has the largest area under the curve (AUC). This indicates that SVM is a better accuracy model at predicting the positive class when the actual outcome is positive.

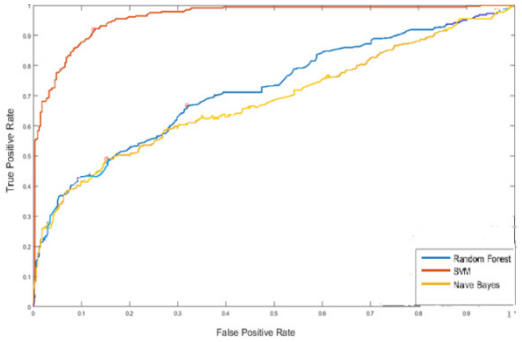

Fig. 4. RoC Graph of Training Data.

B. Mitigation of DoS attack with Intelligent Method of Reinforcement Learning

Reinforcement Learning (RL) refers to a category of Machine Learning method in which the agent receives a reward later, in the next time step while assessing its previous action. Basic reinforcement is modelled *as Markov Decision Process* [5]. Typically, a RL setup is consists of two main factors, an agent and an environment as shown in Fig. 5.

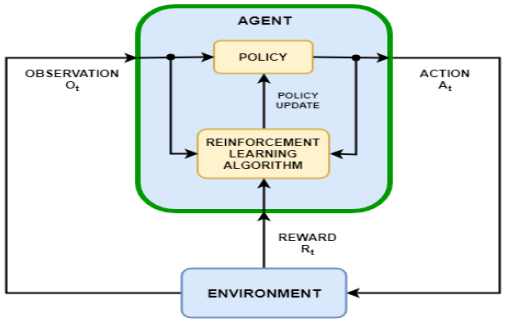

Fig. 5. Components of Reinforcement Learning.

In RL, suitable action is taken to maximize reward in a particular situation. Without training dataset, it will gain from its experience. Apart from the two main elements i.e. an agent and the environment, RL consists of four main subelements: a *policy*, a *reward* function, a *value* function, and a *model* of the environment [16].

The agent and environment communicate with each other sequentially indiscrete time steps, t = 0; 1; 2; 3 [6]. At each time step t, the agent receives some depiction of the environment's state, $S_t \in S$, where S is the set of possible states, and selects a corresponding action on that basis. $A_t \in A(S_t)$, where $A(S_t)$ K is the set of actions accessible in state S_t. One step later, partially as an outcome of its action, the agent gets a numerical reward, $R_{t+1} \in R$, and ends up in another state, S_{t+1}.At each time step, the agent executes a mapping from states to probabilities choosing each possible action. This mapping is known as the agents policy and is expressed as π_t, where $\pi_t(s,a)$ is the likelihood that $A_t = a$ if $S_t = s$. RL techniques determine how the agent changes its strategy because of its experience. The agent's goal, is to maximize the total amount of reward it receives over the long period of time [7].

C. Proposed Novel Algorithm to detect the attacker

A novel algorithm is proposed in mitigation module to reduce our search of attacker machine to a limited number of hosts associated with the switch identified. It makes use of flow based technique [4]. Assuming that with attack packets, the packet flow will increase, hence the idea is to compute the flow for every switch. The advantage of the algorithm is it reduces the number of searches in case of a large network topology. The algorithm is divided into 3 phases:

Initialization: Consider a network of n switches. We initialize a vector of n elements with each element associated with the respective witch. At the beginning of the experiment, we collect flow in each switch, say, $\{f_1, f_2, \ldots, f_n\}$, find the total flow, $F=\sum_{i=1}^{n} fi$,and store the normalized value of flow $nf_i = f_i/F$ for all $i \in [1, n]$. We do this in every epoch. Thus, for every epoch j, we have a vector of normalized flow $V_j = nf_1, nf_2, \ldots, nf_n$. This vector V_j is the input to our attacker isolation algorithm.

Iteration: The normalized flow vector V_j for an epoch j is searched for the element having maximum value. This element happens to be the one associated with a switch recording maximum flow in that epoch. Having identified this switch, a signal is sent to the controller to disable/deactivate this switch [8]. The flow in the remaining switch is checked for few epochs, especially the switch(es) associated with the target machines. This is repeated until the stopping condition is met.

Termination: Each switch under normal traffic flow have some flow value. This is adjusted as per the experimental setup and may be different for different switches in the network. Let this minimum flow value or the target machines be f_{min}. threshold value for the switch s_t associated with the

During the iteration steps, when a switch $s_k$is deactivated, we check the flow in s_t. If the flow in s_t drops below f_{min} we stop the algorithm and mark s_k as the switch having higher probability of having attacker machine associated with it. The proposed algorithm is as stated below:

Algorithm

- **Input:** The normalized flow vector V_j for an epoch j
- **Output:** The switch s_k, as the switch having higher probability of having attacker machine associated with it.

1. Initialize the normalized flow vector V_j for an epoch j
2. Find the element with maximum value in the vector V_j, say nf_k.
3. Send a signal to the controller to deactivate the corresponding switch, s_k
4. Check the flow in switch s_t associated with target machines
5. If flow in $s_t$is less than f_{min} then goto step9.
6. Activate switch s_k
7. Increment j by fewepochs
8. Goto step 2
9. Output the switch $s_k$as the one having higher probability of being associated with attacker machine Stop.

III. RESULT AND DISCUSSION

A. Experimental Test bed Topology

The Network Setup of Fig. 6 is used in our experiments which consists 17 hosts, 6 switches and 1 controller.

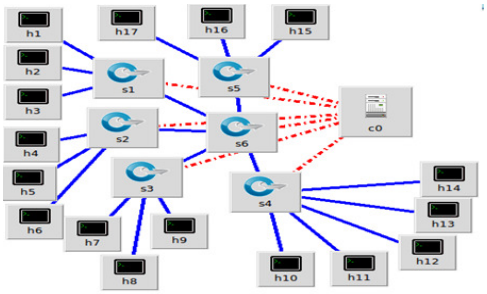

Fig. 6. One of the experimental network setup.

The architecture is as follows: Hosts under switches s1, s2 and s3, the target hosts, are the ones to be protected. Hosts under switch s5 will be used for support in the form of sentinels, honeypots, compute-intensive processing, etc. Controller c0 acts as a forwarding switch and is connected to all switches. A Client-Server Model is used so as to have communication between the agent and the hosts and pass messages of attack traffic. It is also used to send activation/deactivation messages to sentinels, observe patterns of attack if a traffic is redirected to one of the honeypots, provide support functions for compute intensive processing [10].

B. Fuzzy Controller subsystem implementation

To generate the states of our reinforcement learning agent, we propose a fuzzy controller system, that takes in as input various variables of the environment and based on the rules fed into it, generates the state of the agent. For example, if the input variables considered are the priority levels of the service, and number of packets received by a host [14]. The parameter service is a number given to the different networking services available in our system on hosts that are to be provided security. With lower values given to services with low priority. The second parameter could be, say, number of packets received by a host, is identified by the module that is keeping track of the traffic flow and extracting relevant information, cleaning it up and presenting in a format to be used by our controller, is classified into three categories, based on the volume of the incoming traffic seen by a host. Armed with these two parameters, our controller will churn out the state of our agent based on the rules fed into it. The Table 1 presents the set of rules fed to the fuzzy controller based on two parameters, service priority and No. of packets.

Table 1: Set of Rules fed to the fuzzy controller.

Service Priority	Feature (No. of packets)	State of Agent
Low	Low	Low Risk
Low	Medium	Low Risk
Low	High	Medium Risk
Medium	Low	Low Risk
Medium	Medium	Medium Risk
Medium	High	Medium Risk
High	Low	Low Risk
High	Medium	Medium Risk
High	High	High Risk

The security levels for hosts under switches s1, s2 and s3 are defined at various levels depending on the application priority. What it means is that, network services running on hosts under switch s1 is of low risk and low priority and thus the action to be performed in the event of an attack is of minimalist in nature as compared to the hosts under switches s2 and s3. This level will act as one of the parameter to decide the state of our reinforcement learning agent.

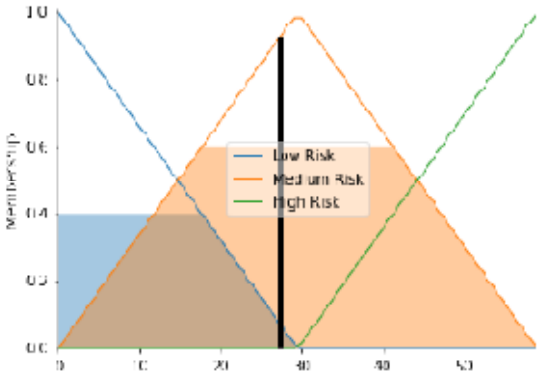

Fig. 7. State of the agent as identified by fuzzy controller.

Based on the number of packets received on a host for a particular service and the level of that service, our controller will give out the state our agent should be in, for that particular episode. This is by far, a very novel and a very efficient way of reading the environment as shown in Fig. 7.

C. Results of Experimental Emulation with 3 Flooding Attacks

As mentioned earlier, our research will be highlighting on various DoS flooding attack, mainly TCP-Syn, ICMP and UDP flood attacks were studied. An experimental test bed with mininet emulator was created [20], as shown in Fig. 6 with hosts connected to switch s1, s2, s3 e.g host h1, which is the victim node and host connected to switch s4 *e.g* host, h10 is the attacker node. First we consider a normal traffic flow as shown in Fig. 8 and its corresponding graph of normal traffic at each switch port in Fig. 9 as analysed by wires hark.

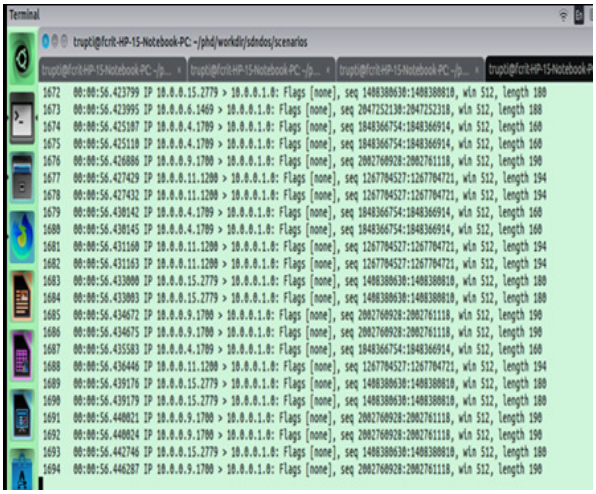

Fig. 8. A snapshot of Normal traffic flow.

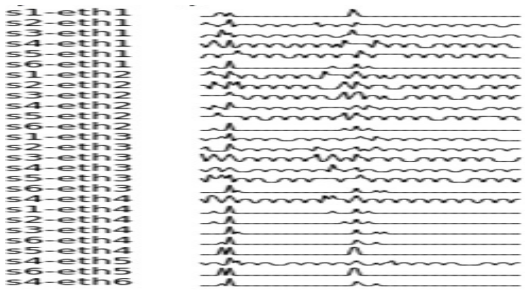

Fig. 9. A snapshot of Normal traffic flow graph outputs.

During normal traffic flow, as seen in the experimentation snapshot of Fig. 9 , there is no much spiked increase in traffic, however in case of the TCP-Syn attack there will be increased spike in traffic as shown in the below given Fig. 10.

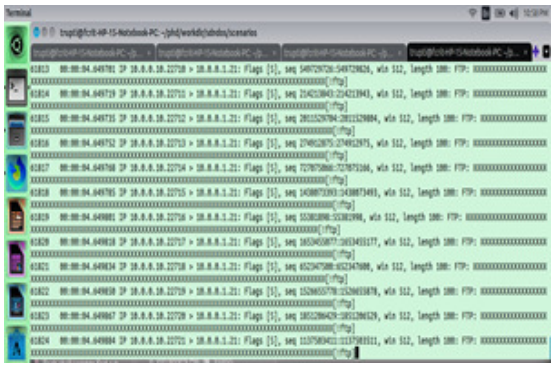

Fig. 10. A snapshot of TCP Syn flood Attack.

Once the state of the Agent is known to change, the corresponding mitigation action is taken the actions defined are for no attack or normal traffic, action performed is accept the packet, for a low attack, we redirect the traffic, for medium attack, we block the packets *i.e* reject the packet and send an error message (ICMP) and for high severity attack we drop the packets and no reply given, so attacker is unaware of drop. The agent will get a reward or a punishment for every correct or wrong action taken respectively. This is known as a positive or negative reward. The agent does self-learning from the environment. The main aim is to find a policy πmaximise the rewards so that agent trains itself to take the correct mitigation action. Fig.11 gives the experimental results of action performed and the reward given to the agent [19].

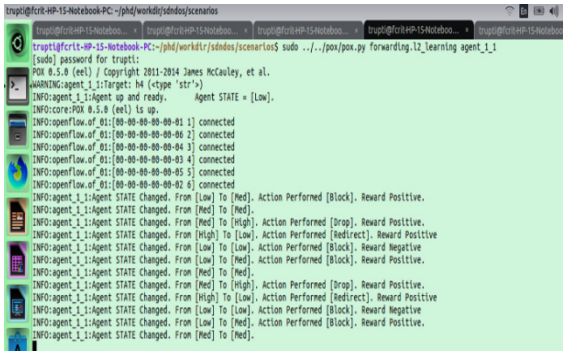

Fig. 11. Snapshot of Mitigation Output Taken and the rewards gained.

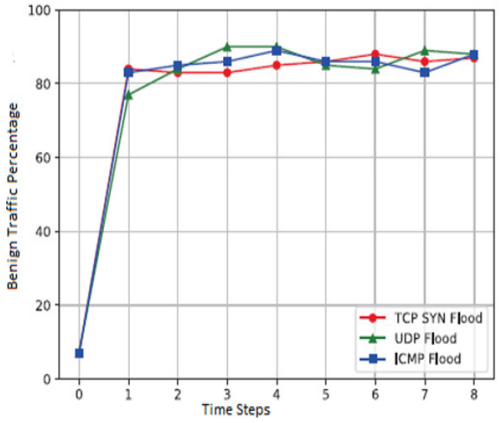

Fig. 12. Snapshot of Benign Traffic flowing inspite of the attack.

An experimentation with mininet emulator [21] proofs the success of mitigation technique by giving higher percentage of benign traffic inspite of the attack traffic. Implementation method proposed as shown in Fig. 12 indicates that the attack is been successfully mitigated since, inspite of the flood attacks like TCP Syn [12], UDP and ICMP traffic, benign traffic keeps flowing, keeping the network working, which was the main motive of the research.

IV. CONCLUSION

In this paper, we have presented a current review on research on SDN security. We have detected the DoS attack in SDN using Support Vector Machine (SVM).
We have also proposed an intelligent method of Reinforcement Learning which supports a Morkovian Mathematical Model for mitigation of DoS attacks in SDN.
The Model is trained with Fuzzy Logic. An experimental test bed is designed with systems and 3 types of flooding attacks i.e. TCP-Syn, ICMP and UDP flood attacks. The results also show that with mitigation, the rate of benign traffic has increased indicating a high level of accuracy with respect to benign traffic, allowing the network to work uninterrupted, inspite of the attack traffic. This indicates the successful implementation of the proposed methodology framework.

V. FUTURE SCOPE

The framework created is tested for the flood category of DoS attack e.g UDP, ICMP etc. however other DoS attacks for *e.g.* Man -In -Middle attack could also be tested with this framework. The mitigation technique is implemented with MiniNet which is a Network emulator which can be later tested real time on SDN Switches.

Conflict of Interest. Nil.

REFERENCES

[1]. Kalkan, K., Gür, G., & Alagöz, F. (2017). SDNScore: A statistical defense mechanism against DDoS attacks in SDN environment. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, *12*(1), 669-675.
[2]. Scott-Hayward, S., Natarajan, S., & Sezer, S. (2015). A survey of security in software defined networks. *IEEE Communications Surveys & Tutorials*, *18*(1), 623-654.

[3]. Zargar, S., Joshi, J., & Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE communication survey and tutorials, 15*(4), 2046–2069.
[4]. Shin, S., Yegneswaran, V., Porras, P., & Gu, G. (2013). 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.
[5]. Wang, H., Xu, L., & Gu, G. (2015). Floodguard: A dos attack prevention extension in software-defined networks. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 15*(1), 239–250
[6]. Van Trung, P., Huong, T. T., Van Tuyen, D., Duc, D. M., Thanh, N. H., & Marshall, A. (2015, October). A multi-criteria-based DDoS-attack prevention solution using software defined networking. In *2015 International Conference on Advanced Technologies for Communications (ATC)* (pp. 308-313). IEEE.
[7]. Sezer, S., Scott-Hayward, S., Chouhan, P. K., Fraser, B., Lake, D., Finnegan, J., & Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), 36-43.
[8]. Dhawan, M., Poddar, R., Mahajan, K., & Mann, V. (2015). SPHINX: Detecting Security Attacks in Software-Defined Networks. In *Ndss*, *15*, 8-11.
[9]. Wang, H., Xu, L., & Gu, G. (2014). OF-GUARD: A DoS attack prevention extension in software-defined networks. *The Open Network Summit (ONS)*, (2014).
[10]. Charu, P. P., & John, M. (2016). A Framework for Design and Simulation of DoS attacks on SDN Network. *international journal of innovative research in computer and communication engineering*, *4*(2), 345-356.
[11]. Wang, H., Xu, L., & Gu, G. (2015). Floodguard: A dos attack prevention extension in software-defined networks. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 13*(2)*,* 239–250.
[12]. Mohammadi, R., Javidan, R., & Conti, M. (2017). Slicots: An sdn-based lightweight countermeasure for tcp syn flooding attacks. *IEEE Transactions on Network and Service Management*, *14*(2), 487-497.
[13]. Bhandari, A., Sangal, A. L., & Kumar, K. (2015). Destination address entropy based detection and traceback approach against distributed denial of service attacks. *International Journal of Computer Network and Information Security*, *7*(8), 9-20.
[14]. Wang, R., Jia, Z., & Ju, L. (2015). An entropy-based distributed DDoS detection mechanism in software-defined networking. In *2015 IEEE Trustcom/BigDataSE/ISPA, 15*(1): 310–317.
[15]. Liu, Y., Dong, M., Ota, K., Li, J., & Wu, J. (2018). Deep reinforcement learning based smart mitigation of DDoS flooding in software-defined networks. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (pp. 1-6). IEEE.
[16]. Sharma, S., Sahu, S. K., & Jena, S. K. (2015). On selection of attributes for entropy based detection of DDoS. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI),* 1096-1100.
[17]. Feng, W., Zhang, Q., Hu, G., & Huang, J. X. (2014). Mining network data for intrusion detection through combining SVMs with ant colony networks. *Future Generation Computer Systems*, *37*(2), 127-140.
[18]. Da Silva, A. S., Machado, C. C., Bisol, R. V., Granville, L. Z., & Schaeffer-Filho, A. (2015, September). Identification and selection of flow features for accurate traffic classification in SDN. In *2015 IEEE 14th International Symposium on Network Computing and Applications*, 134-141.
[19]. Kokila, R. T., Selvi, S. T., & Govindarajan, K. (2014). DDoS detection and analysis in SDN-based environment using support vector machine classifier. In *2014 Sixth International Conference on Advanced Computing (ICoAC),* 205-210.
[20]. M. Team, “Mininet - an instant virtual network on your laptop (or other pc).” [Online]. Available: http://mininet.org/
[21]. M. MC, “Nox.” [Online]. Available: https://github.com/noxrepo/pox.

How to cite this article: Lotlikar, T. and Shah, D. (2020). A Novel Hybrid Mitigation Technique against DoS Attacks in Software defined Network with Entropy, SVM and Reinforcement Learning. *International Journal on Emerging Technologies*, *11*(2): 627–632.