ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 6, Issue 4, April 2021

DDoS Attack Detection Using OpenDaylight in Internet of Things (IoT)

[1] Omerah Yousuf, [2] Roohie Naaz Mir

[1][2] Department of Computer Science and engineering, NIT Srinagar, Hazratbal, Jammu & Kashmir, India
Email: [1]omerahyousuf@nitsri.net, [2] naaz310@nitsri.net

***Abstract*— Internet of Things (IoT) is a forthcoming analysis field and is being thought to be the revolution within the world of communication as a result of its prominent applications in various fields. These networks are highly vulnerable to attacks due to their transparent and self-assimilation nature. As a consequence, security is the highest concern in this framework and requires new types of strategies to be developed to counter different types of attacks in IoT. The aim of a Denial of Service (DoS) attack is to make server services inaccessible to the intended user, and a DDoS attack occurs when multiple DoS attacks are present in a network. As distributed denial of service (DDoS) attacks become more popular on the Internet, solutions to combat them are in greater demand. In this paper, our strategy is to utilize OpenDaylight as an SDN controller to find DDoS attacks in IoT networks by applying the Support Vector Machine (SVM) approach. Here we tend to create a Software-defined Network (SDN), that consists of 20- Devices, 4-Open Flow switches, and an Open Flow controller (Opendaylight controller). The nodes/Devices are directly connected with switches. Our technique is to find DDoS assailant nodes victimization SVM by classifying the malicious and non-malicious data in the network. The detection rate for this SDN controller is remarkably very efficient as compared to the other SDN controllers.**

***Keywords*— Internet of Things, DDoS attack, OpenDaylight, Support Vector Machine**

I. INTRODUCTION

Internet of things (IoT) is a paradigm shifting, imminent model in the subject of wireless communications. In layman's terms we can define IoT as a wireless system of interrelated objects that include devices like RIFD (Radio Frequency Identification Tags), sensors, mobiles, digital machines, animals or even people that are provided with UIDs (Unique identifiers) & have the capability to communicate over a network without the need of human to human computer guidance/interactions [1]. The applications of IoT implementations are vast & ever increasing. Some of the technologies that make heavy use of IoT include public security, infrastructure development, connected health, smart homes, smart cities, smart grids, wearable devices, agriculture, industry automation, business services etc. IoT has the potential to integrate production & service management & also to an extent integration of our physical world with the digital realm; which has been the end-goal feature desired in almost every type of interactive device or a service. Amongst the most significant characteristics of IoT is its ability to "Self-configure". IoT networks consists of several nodes & devices like sensors, actuators etc, configuration of such a system by a hand to hand method is complex but IoT can handle it efficiently, configuring as per the need of users and applications. This all translates to proper automated inter-connection between various devices making up the network system. To achieve this goal of self-configuration unique addressing & communication components are leveraged [2].

The primary technological hurdles & problems while actualizing the idea of IoT is that wide range of gadgets/devices should be broadly used with wide acceptance, therefore giving inter-operability between them. They should have capability of adjusting to various different network systems during which their autonomous behavior should be maintained. Simultaneously, three major factors such as trust, security and privacy of the device connected to the network should be upheld. Since these networks are autonomous in nature, and performs auto-configuration of new device when entering into the network, this leads to open nature of IoT. However openness of IoT also makes it vulnerable to security attacks. These attacks may aim at denying the services provided by IoT or to take over the whole network. The approach attackers take can be varied, making use of variety of security attacks & exploit including physical attacks, eavesdropping etc. However, the most challenging attack for an IoT system would be DDoS attacks (Distributed denial of service). This attack overloads a network with repeated false requests with the aim of degrading the performance of the system, ultimately causing it to fail. If successful this would cause denial of services being provided by the network i.e., IoT here, to its legitimate users [3]. Distributed in DDoS refers to the fact

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 6, Issue 4, April 2021

that a DOS attack is being executed by multiple attackers/agents on the network from various locations. Any IoT network will only have limited resources at its disposal. When a DDoS attack is executed on a network, it would start allocating its resources for serving the requests. However, when the number of requests increases above the threshold a network can handle, it won't be able to serve any more requests. At this point any request even if made by legitimate users would be denied & hence cause the disruption in the delivery of services being provided by IoT [4].

Many researchers have been extensively interested in designing SDN-based network security solutions as a result of recent advances in software-defined networking (SDN) and its rapid and high acceptance in the network community. After their adoption in large-scale wide area networks SDN-based solutions have gotten more recognition [5].

Many distinct features of SDN are essential in detecting and mitigating DDoS attacks. Separation of the control plane from the data plane, a logically centralized controller, network programmability by external applications, software-based traffic analysis, and the ability to dynamically change forwarding rules are among these features [6].

In this paper, we add to past studies by proposing a new method for detecting DDoS attacks in the Internet of Things by using OpenDaylight as an SDN controller to build a topology with 20 hosts, four OpenFlow switches, and an OpenFlow controller. Based on the threshold values assigned to each node, we used a Support Vector Machine classifier to identify the host as Normal or Attacker node.

The following is a breakdown of the paper's structure. Section II provides an overview of OpenDaylight as an SDN controller. Section III presents the related work. The proposed SVM-based DDoS attack detection algorithm using the OpenDaylight SDN controller is presented in Section IV. The experiments and results are described in Section V. Our conclusion and references are enumerated in this paper..

II. OPENDAYLIGHT AS SDN CONTROLLER

OpenDaylight is an open source framework for Software Defined Networking (SDN). Many other SDN controllers, such as Ryu, Pox, and others, were also designed in the past. As part of its framework, OpenDaylight also supports OpenFlow and provides ready-to-install network solutions [7].

The ODL controller is entirely software-based and runs in its own Java Virtual Machine (JVM). As a result, it can be installed on any Java-enabled hardware and operating system. Figure 1 depicts the architecture of OpenDaylight SDN Controller [8].

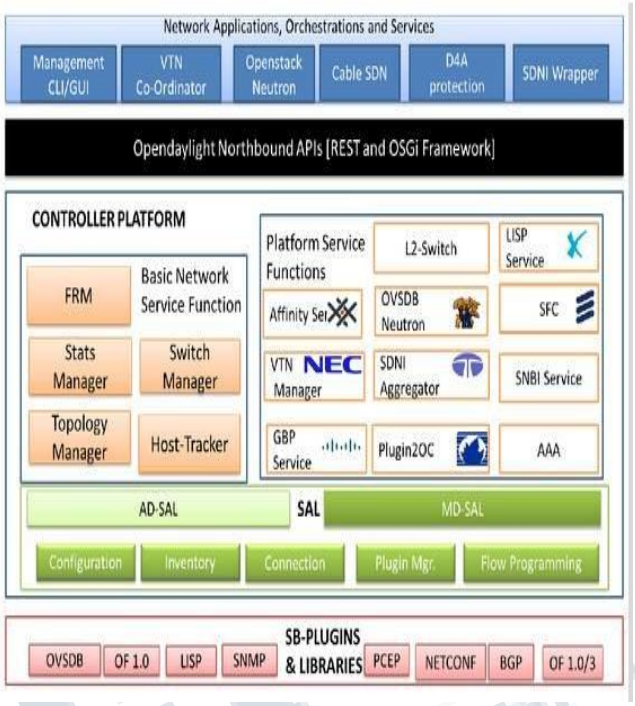

Figure 1: Architecture of ODL

There are multiple levels to the OpenDaylight SDN Controller. Business and network logic applications make up the top layer. The system layer is in the middle, and the physical and virtual devices are at the bottom. The context in which the SDN abstractions will manifest is the middle layer. This layer contains APIs that are both northbound and southbound. The controller exposes open northbound APIs that applications can use. For the northbound API, OpenDaylight supports the OSGi (Open Services Gateway initiative) system and bidirectional REST. Above the middle layer, the applications include the business logic [9].

Since ODL was developed with the aim of reducing vendor lock-in, it supports protocols other than OpenFlow. Multiple protocols, such as OpenFlow and BGP-LS, can be supported as separate plugins on the southbound interface. No matter what the controller's basic protocol, and the network equipment is, the Service Abstraction Layer (SAL) defines how to provide the requested service [10].

III. RELATED WORK

Due to the advancement in network technology, DDoS attacks are becoming more complicated. Typically, network traffic must be constantly monitored and analyzed to prevent authorized users from breaching network policies and to protect them from various types of attacks. In the recent past, there has been a lot of research performed in the field

of network security in IoT by well-known authors. Since it is one of the most common research areas in the 21th century, we can easily find a large amount of research material on which to base this paper and suggest a new mechanism that will be more effective than current approaches.

The authors in [11] proposed Learning Automata (LA) ideas to defend against the DDoS attacks in IoT networks. Here, the authors used Service Oriented Architecture (SOA) as a framework model for IoT, acting as a middleware and providing opportunities for developers to build applications. This model not only provides IoT facilities, but also ensures that the entire system is protected from DDoS attacks. The authors of this paper used a cross-layer model to counter DDoS attacks at any level, making it compatible with a wide range of artefacts. After discovering the DDoS attack in the IoT, the required measures are taken to restrict the number of incoming requests from the attacker. This scheme is very successful at preventing DDoS attacks in the IoT by maximising resource utilisation.

The authors in [12] suggested SDN as an efficient method to identify various DDoS attacks in IoT. Here, the authors proposed a sequential and concurrent approach for detecting possible and suspicious victims in the network using the SDN's flow monitoring capability. For high detection precision, these methods were used to capture both the flow volume and the flow rate asymmetry features. To capture these two characteristics, developers recorded the total flow rate coming in and out of any possible victim IP while keeping in mind TCAM's scale. Both strategies will try to keep IP ranges as limited as possible during the start-up phase. If you need to locate a victim quickly, the sequential approach is better whereas the concurrent method is better for finding both the victim and the attackers quickly. However, these approaches need to be evaluated in the real world and on the OpenFlow platform in the future.

The authors in [13] proposed a naive Bayes classifier with two frequency based methods (i) Discrete Fourier transform (DFT) and (ii) Discrete wavelet transform (DWT), thereby differentiating between malicious and non-malicious traffic using coefficients. The wavelet transform provides higher resolution information in the frequency domain, which improves detection accuracy. When compared to other classifiers, the Naive Bayes classifier is quick, simple to implement, and effective at classifying attack and normal traffic. As compared to DWT functionality, the device performs better when DFT attributes are used.

The authors in [14] introduced an intrusion detection system which utilizes Support Vector Machine (SVM) classifier to detect DDoS attacks in IoT, with high accuracy, less false positive rate and accurate classification as compared to other classifiers and the experiments conducted using DARPA dataset. The authors here intend to combine the traffic pattern built into SVM with the SDN controller in order to detect DDoS attacks in real time. SVM, on the other hand, may take longer to train and generate the detection model, which is then used to predict network traffic characteristics. The efficiency of the SVM classifier can be greatly improved by combining AVL tree and SVM and the testing time can be reduced comparatively by using the height balancing property of an AVL tree.

The authors in [15] suggested an innovative algorithm popularly known as IP Address Interaction Feature (IAI) algorithm based upon source address interaction of normal flow. The learning of margin based SVM classifiers depends upon learning samples collected from both normal and flow attacks, which leads to identification of current network flow and DDoS attack flows. This method shows effective performance in identification of unusual phenomenon introduced by DDoS attack flows, especially when attacking traffic is masked by huge number of normal flows. The method has lesser number of false positives and is computationally time efficient as compared to other existing methods.

IV. PROPOSED DDOS DETECTION ALGORITHM

DDoS attacks in IoT have been proposed by the researchers from time to time. Software Defined Networking (SDN), on the other hand is a modern network management technology that is attracting a lot of interest from research and business. According to research, the frequency of DDoS attacks has been increasing in recent times. As a result, one of the major challenges in network measurement is how to efficiently and rapidly detect DDoS attacks. SDN is an excellent platform for DDoS detection since the central controller can easily install and adjust measurement rules on all switches in a coordinated manner. A prominent role in our work is to investigate and choose a suitable OpenFlow controller. OpenDaylight, a Java- based open source controller has been chosen for detecting the attacker node in the network using Support vector machine (SVM) approach. The proposed algorithm is based on classical problem solving approach and works on the concept of detection of DDoS attacker node using a pre-defined threshold value.

The ODL based DDoS Detection method is given below:

Algorithm 1: Detection of DDoS attack in an OpenFlow environment (ODL) using SVM approach;

Input: n: Number of nodes (k1, k2……kn)
X=20: Number of devices connected to the network (SDN)
Y=4: Number of OpenFlow

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 6, Issue 4, April 2021

Switches (S1,S2,S3,S4)
Z: OpenFlow controller (ODL)
Threshold, λ=0.5
Output: Data Aggregation Tree DAT (V, E)
1: procedure Detecting DDoS AttackerNode (n, X, Y, Z, λ);
2: Create a Software-defined Network with parameters as X=20, Y=4 and Z=1;
3: Connect (k1, k2……kn) with (S1, S2, S3, S4) directly;
4: Choose ks as Source Node and kd as destination node;
5. Based on deep learning approach, assign each node a priority (pi);
6. Transfer the data between the nodes, switches and other devices with {SIP, Po, Pi, DIP)
Where SIP = Source IP;
Po = Port;
Pi = Priority;
DIP = Destination IP;
7. for each Node (i=1 to n)
{
8. Classify the Node data using SVM approach to detect DDoS attacker Node;
{
9. if data[ni]> λ
10. Classify [ni] as Normal node;
else
11. Classify [ni] as DDoS attacker node;
}
12. end for;
13. end procedure;

Since the nodes in the ODL environment are connected to each other as well as the openflow switches which are controlled by a common ODL controller, the nodes will receive the data at the rate proportional to the size of the network. While running the simulations, we observe the input data rates at each and every node. Whenever there is no attack, the simulations works perfectly with almost all the nodes having a comparable data entry rate. From the simulations over time and again, we can find out the maximum value of the data input rate at each node. Based on the history of the simulations carried out and the work of some eminent researchers in the field of SDN, it has been concluded that the threshold value needs to be set to the value 0.5.

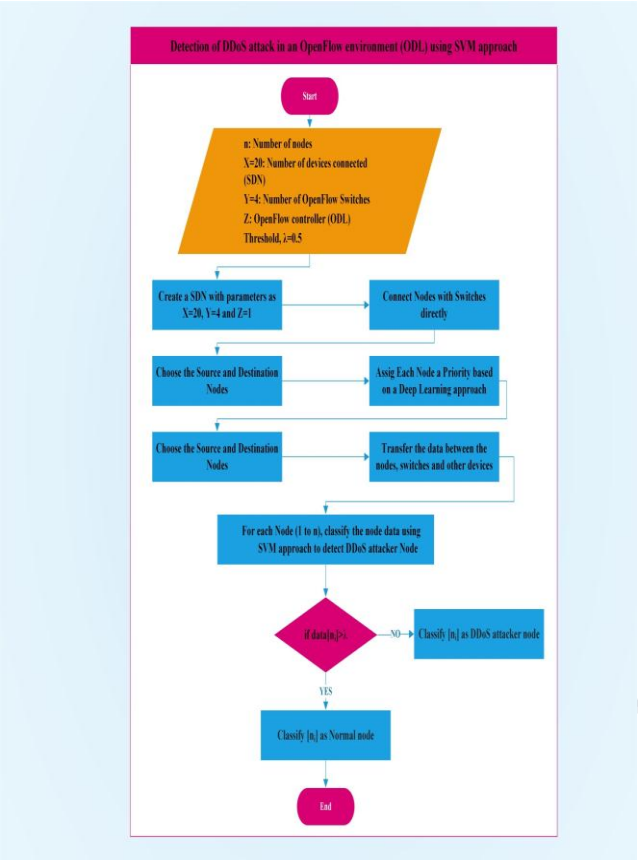

Figure 2: Flowchart for DDoS Detection using ODL

We exploit the same property of the threshold value being 0.5 to detect the DDoS attacker node. We can easily conclude this that during the normal data transmission between the nodes, switches and the ODL controller, the value of threshold never exceeds 0.5 at any node.

However, when the attacker node is present in the network, and launches a DDoS attack on the network, the node responsible for launching the DDoS attack has the value of the threshold value greater than 0.5. This way we can conclude that the attacker node is present in the network and can be detected as well since the data transmission between the nodes and the switches occurs using the parameters:

{SIP, Po, Pi, DIP)
Where SIP = Source IP;
Po = Port;
Pi = Priority;
DIP = Destination IP;

Hence by this algorithm, not only we can conclude that an attacker node is present in the network, but can also detect it as well. The flowchart depicting the working of the proposed algorithm is shown in Figure 2.

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 6, Issue 4, April 2021

V. EXPERIMENTAL RESULTS AND DISCUSSIONS

In order to implement the above described DDoS detection algorithm, the experiments in this work are implemented using OpenDaylight (version Beryllium) controller for creating the SDN network topology on an Ubuntu 14.04 VMware along with the java integration. Our VMware is implemented using dual core processor with 2GB or above of RAM. Here we created a simulation based process only and does not used any real time data sets or any realtime environment for experimentation. Our SDN testbed consists of 20 hosts (k1 to k20), 4-Open Flow Switches (S1 to S4) and one Open Flow controller (OpenDaylight controller) as shown in Figure 3.

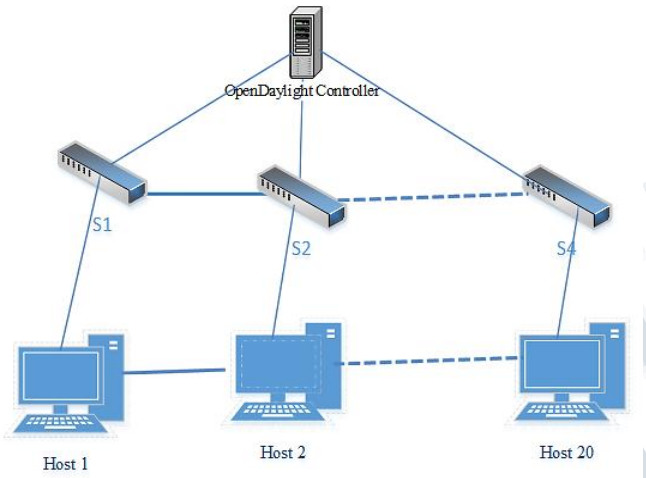

Figure 3: SDN Topology

Next step is to select the source and destination Node from the created network topology and allocate the node scheduling priority based on the deep learning approach by using WARMUP_ITERATIONS along with the scheduling timeUnit based on the MILLISECONDS as shown in Figure 4.

```
public class IoT_SDN {
    private static final int WARMUP_ITERATIONS = 20;
    private static final int MEASUREMENT_ITERATIONS = 20;
    private static final int OUTER_LIST_20 = 20000;
    private static final int NoofDevice = 20;
    private static final int Noofswitch = 4;
    private static final int Noofcontroller = 1;
      @Warmup(iterations = WARMUP_ITERATIONS, timeUnit = TimeUnit.MILLISECONDS)
    @Measurement(iterations = MEASUREMENT_ITERATIONS, timeUnit = TimeUnit.MILLISECONDS)
    public void Processwith20Devices04SwitchTree() throws DataValidationFailedException
        for (int outerListKey = 0; outerListKey < Noofcontroller; ++outerListKey) {
          System.out.println("Controller Created");
        }
        for (int outerListKey = 0; outerListKey < Noofswitch; ++outerListKey) {
          System.out.println("Switch Node Created");
```

Figure 4: Screenshot of Warmup Iterations

After this step, we transferred the data from source IP address to destination IP address within this network, for example, here from 192.168.1.080 (Source) to 192.168.1.180 (Destination).

Finally, the data is classified by using SVM java classifier and detected the DDOS attacker node in the network based on the threshold value of 0.5 as shown in Figure 5.

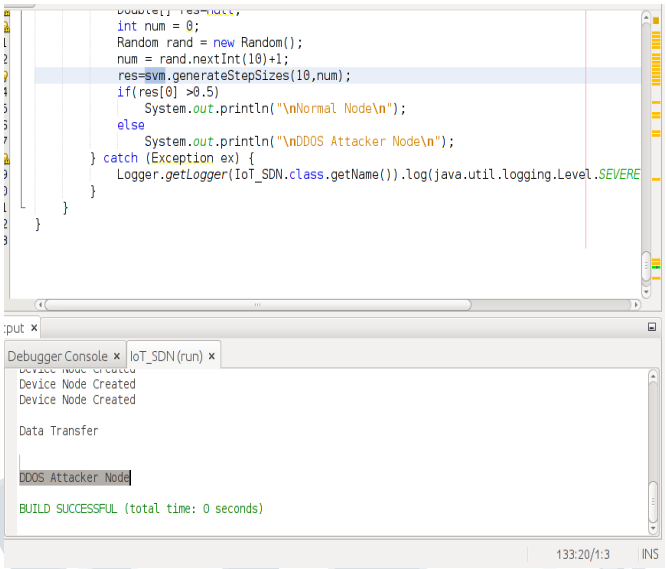

Figure 5: Screenshot of Data Classification

Here the threshold value for each node was assigned by using java.util.Random class and then invoked the method nextInt ( ) by creating the instance of that class. The arguments were then passed to the method for setting an upper bound on the number range to be generated. NextInt(10), for example, will generate numbers in the range 0 to 10, both inclusive and exclusive. Once the threshold value for each node has been assigned, all those nodes having the threshold value lesser than 0.5 will be considered as a DDoS **Attacker node** while those having greater than 0.5 threshold value will be considered as **Normal node.**

VI. CONCLUSION

IoT) is a hot topic these days, and the number of devices connected is growing rapidly, resulting in an increase in the number of sources from which DDoS attacks can be launched. In this paper, we examined at how to employ OpenDaylight (SDN) for detecting DDoS attacks using Support Vector Machine (SVM) classifier. We proposed an algorithm for DDoS attack detection in IoT by creating a SDN in OpenDaylight and then transferred the data from the Source to Destination node in the network. In this method, priorities were allocated to each node using WARMUP_ITERATIONS and then finally classified the data using SVM and detected the attacker node in the network based on threshold value assigned to each node.

Here the simulations were performed using the java packages and does not used any real time data sets or any realtime environment. The detection rate of this method by using OpenDaylight was significantly better compared to the other SDN controllers.

REFERENCES

[1] Jia, Yizhen, Fangtian Zhong, Arwa Alrawais, Bei Gong, and Xiuzhen Cheng. "Flowguard: an intelligent edge defense mechanism against IoT DDoS attacks." IEEE Internet of Things Journal 7, no. 10 (2020): 9552-9562.

[2] Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." Computer networks 54, no. 15 (2010): 2787-2805.

[3] Mirkovic, Jelena, and Peter Reiher. "A taxonomy of DDoS attack and DDoS defense mechanisms." ACM SIGCOMM Computer Communication Review 34, no. 2 (2004): 39-53.

[4] Asosheh, Abbass, and Naghmeh Ramezani. "A comprehensive taxonomy of DDOS attacks and defense mechanism applying in a smart classification." WSEAS Transactions on Computers 7, no. 4 (2008): 281-290.

[5] Bawany, Narmeen Zakaria, Jawwad A. Shamsi, and Khaled Salah. "DDoS attack detection and mitigation using SDN: methods, practices, and solutions." Arabian Journal for Science and Engineering 42, no. 2 (2017): 425-441.

[6] Yan, Qiao, and F. Richard Yu. "Distributed denial of service attacks in software-defined networking with cloud computing." IEEE Communications Magazine 53, no. 4 (2015): 52-59.

[7] Khattak, Zuhran Khan, Muhammad Awais, and Adnan Iqbal. "Performance evaluation of OpenDaylight SDN controller." In 2014 20th IEEE international conference on parallel and distributed systems (ICPADS), pp. 671-676. IEEE, 2014.

[8] Arbettu, Ramachandra Kamath, Rahamatullah Khondoker, Kpatcha Bayarou, and Frank Weber. "Security analysis of OpenDaylight, ONOS, Rosemary and Ryu SDN controllers." In 2016 17th International telecommunications network strategy and planning symposium (Networks), pp. 37-44. IEEE, 2016.

[9] Rowshanrad, Shiva, Vajihe Abdi, and Manijeh Keshtgari. "Performance evaluation of SDN controllers: Floodlight and OpenDaylight." IIUM Engineering Journal 17, no. 2 (2016): 47-57.

[10] Asadollahi, S., B. Goswami, and A. M. Gonsai. "Implementation of SDN using OpenDayLight controller." International Journal of Innovative Research in Computer and Communication Engineering 5, no. 2 (2017): 218-227.

[11] Misra, Sudip, P. Venkata Krishna, Harshit Agarwal, Antriksh Saxena, and Mohammad S. Obaidat. "A learning automata based solution for preventing distributed denial of service in internet of things." In 2011 international conference on internet of things and 4th international conference on cyber, physical and social computing, pp. 114-122. IEEE, 2011.

[12] Xu, Yang, and Yong Liu. "DDoS attack detection under SDN context." In IEEE INFOCOM 2016-the 35th annual IEEE international conference on computer communications, pp. 1-9. IEEE, 2016.

[13] Fouladi, Ramin Fadaei, Cemil Eren Kayatas, and Emin Anarim. "Frequency based DDoS attack detection approach using naive Bayes classification." In 2016 39th International Conference on Telecommunications and Signal Processing (TSP), pp. 104-107. IEEE, 2016.

[14] Kokila, R. T., S. Thamarai Selvi, and Kannan Govindarajan. "DDoS detection and analysis in SDN-based environment using support vector machine classifier." In 2014 Sixth International Conference on Advanced Computing (ICoAC), pp. 205-210. IEEE, 2014.

[15] Cheng, Jieren, Jianping Yin, Yun Liu, Zhiping Cai, and Chengkun Wu. "DDoS attack detection using IP address feature interaction." In 2009 International Conference on Intelligent Networking and Collaborative Systems, pp. 113-118. IEEE, 2009.