Grenze International Journal of Engineering and Technology, June Issue

Investigating Parameters of Software Defined Networks for Detecting DDoS Attacks

Amey S. Gawde[1], Ameya K. Naik[2] and Kiran V. Ajetrao[3]
[1-3]K. J. Somaiya College of Engineering, Electronics and Telecommunication Engineering Department, Somaiya Vidyavihar University, Mumbai, India
Email: gawde.as@somaiya.edu, ameyanaik@somiaya.edu, kiranjetrao@somiaya.edu

***Abstract*—In any communication network, for regular maintenance the service provider, needs to manage individual device by login into the specific device. This can be avoided by using Software Defined Networks which gives easy access and the facility of core network management to service provider. This is achieved by separating Control and Data plane of the hardware devices. Using SDN Controller managing the devices is simpler, but it is exposed to attacks making the system vulnerable. The most prominent attacks for SDN are DDoS attack and ARP Spoofing attack. Detecting these attacks at early stage is necessary for every communication network. This paper focuses on designing a method to detect DDoS attack by investigating various network parameters. It is observed that it is possible to detect DDoS attack by monitoring specific parameters CPU process, CPU system, number packets, etc. An effective mitigating strategy can be devised based on these parameters.**

***Index Terms*— SDN, Security, DDoS, attacks.**

I. INTRODUCTION

The infrastructures in Information and Communication Technology (ICT) are expanding fast with respect to the number devices and applications in Internet of Things. When such network topologies are created the topologies become very complex and managing the devices becomes difficult. Software Defined Networking can help in managing the complex network and to transform the complex architecture into a manageable network.

The traditional network referring Fig. 1 has many hardware components in it. The packets need to travel all these devices before reaches the end host. It has several devices like, in Fig 1, the switches, routers, firewalls, which are vertically arranged. This architecture is very complex to manage. The hardware devices are vendor specific so to implement new ideas on these devices are also limited. If we can transform this or map this with the Software Defined Network architecture, then SDN architecture is shown in Fig. 2.

SDN is considered as hardware independent which supports the control of the networking devices remotely irrespective of the vendor. This is an added advantage of using SDN. So, the programming can be done as per the designing from the user point of view. SDN architecture divides the normal hardware device operation into three planes namely application, control and data plane as shown in Fig. 2. The communication between the control plane and the Data plane is called as the South bound Communication and the communication of the controller and the application plane is called as the North Bound Communication. The North bound communication is possible using the Application Program Interfaces (APIs) and the South bound communication is possible using the Open Flow Protocol.

Grenze ID: 01.GIJET.10.2.649

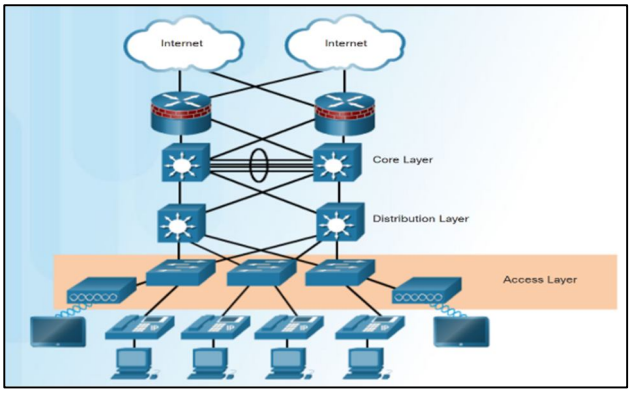

Figure 1. Architecture of Traditional Network

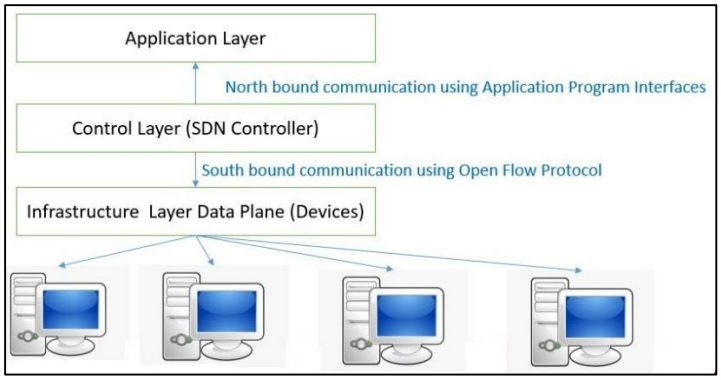

Figure 2. Architecture of Software Defined Network

The devices then become only the data forwarding entity since the entire control part is removed from the traditional hardware devices and controlled by the central controller [1]. In traditional hardware the systems are required to be reconfigured to upgrade it whereas in SDN only the software is required to be updated [2]. This is an advantage in upgrading the system with reduced cost. Programming of SDN is possible using a high-level programming language. This feature helps the users to modify the network parameters based on the operation environment which can help to improve the network performance. SDN provides security, energy efficiency and network function virtualization for improving the performance of the network. SDN can be vulnerable to different types of attacks since the communication between the control and data plane is separated from the hardware.

Providing security to SDN is more challenging as compared to the traditional networking. Different types of attacks are possible on SDN [3] which can be any of the types of namely anomaly attacks, Distributed Denial of Service (DDoS) attacks, intrusion attacks, Denial of Service (DoS), etc., The energy consumption of the hardware devices is increasing as number of devices are increasing. SDN can be used to reduce the energy consumption and provide security to it [1]. These attacks degrade the performance of the network [2]. DoS or DDoS can be performed in many ways like sending flood approaches using TCP SYN, ICMP or UDP packets and host-based approaches, where in more than one host is targeted using specific applications to flood the controller. This may exploit the memory structure, authentication protocol or algorithm.

The paper will be divided into different sections. Section II will reflect advancement in the field of SDN by various researchers. The work done in DDoS detection and mitigation in the SDN environment, which led to the creation of a secure SDN framework, will be summarized in Section III. Section IV will put light on the test bed setup and implementation of DDoS attack. Section V, a DDoS attack detected, various parameters are measured and tabulated for normal and attack scenario.

II. SDN Background and Related Research Work

The idea of decoupling the control and data plane came up after its implementation in telephone networks. The idea of the telephone network implemented in SDN came up with dividing the control and data plane in SDN.

The programming of control plane is hence possible using SDN remotely. SDN also helps the management of devices simple as compared to the traditional network as it can be done in the remotely which is not possible with traditional network. Implementing user specific ideas, entirely as they are, on the hardware network is difficult because of vendor ownership [1]. SDN architecture consists of three different layers namely Infrastructure/data, control, and Application layers. The separation of control and data plane is advantageous but at the same time introduces the security concern in the communication between the control and data plane. The data plane just acts as a forwarding mechanism in case of SDN whereas entire control action is with control plane.

Infrastructure/Data Layer: Switches, routers and access points are the part of infrastructure layer or devices in the data layer. E.g., Open vSwitch, Indigo, Nettle, and hardware switches exist in this layer [4][5]. The function of infrastructure layer is to only forward data packets as per flow rules received from the controller.

Control Layer: It is an important part of the SDN controller. It acts as an intermediary between the application layer and the data layer. All the decisions in accepting or rejecting packets are programmed at the control layer. Routing and flow control is done in control layer. Control layer and infrastructure layer communicate using OpenFlow, Netconf [5] protocols.

Application Layer: It is responsible for handling the software related issues and security applications. Different types of applications handled by application layer include Network Virtualization, Intrusion Detection system (IDS), Intrusion Prevention System (IPS). The communication between controller and the application layer is called as north bound communication and it is possible using Application Program interfaces.[3]

III. DDoS Detection And Mitigation Related Work

Although they can take many various forms, distributed denial of service attacks all aims to overwhelm the network. DDoS attacks are even capable of overwhelming a network and preventing it from continuing to operate. TCP SYN, UDP, and ICMP packet flooding is how the attack is carried out [6]. Attacker on the compromised host will attempt to send packets via switch to the victim host. In response to requests, the controller updates the flow table. The network is instantly shut down when the request is approved because the attacker's goal was to overwhelm the victim with traffic. A selection of the several DDOS attack mitigation strategies that have been studied will be covered in this section.

The authors of paper [7] talked about emulating an SDN network and using sFlow and Mininet for real-time DDoS attack detection and mitigation.

Authors of the research [8] employed Shannon's entropy equation to measure the amount of unpredictability in a network, or entropy. The DDoS attack affects the entropy values.

Ingress, egress, and pushback are defense techniques that are suggested by the authors of article [9] and are used to verify the authenticity of packets in DDoS attacks. By using pushback for one domain and ingress filtering for the same domain, the attacker is found. Machine Learning is deployed for detection and mitigation of attacks.

The idea of entropy is employed by the authors of paper [9] to determine if the network flow is irregular. The neural network technique BiLSTM-RNN is utilized to train the dataset thereby identifying DDoS in real-time and reducing the controller overheads.

The authors of paper [10] talked about emulating an SDN network and using sFlow and Mininet for real-time DDoS attack detection and mitigation. Various DDoS attacks are comprehensively studied in the paper. In the publication [11], the authors employed Shannon's entropy equation to create an algorithm for measuring entropy, or network unpredictability. The DDoS attack affects the entropy values. The POX controller is used for setup. The authors of paper [11] have combined the use of support vector machines with neural networks. These ideas are applied to the detection and categorization of denial-of-service (DDOS) assaults in communication networks. The project is implemented using NS-2.

IV test bed setup and implementation of DDoS attack

The test bed setup for the implementation is done in three steps. In first step a topology is created in ubuntu desktop using mininet. In second step a topology is integrated with sFlow-RT to capture the packets graphically. In the last step a RYU controller is used to which will show the flooding of packets and in the future scope will be designed with blockchain technology to mitigate the attack.

An Ubuntu 20.04 operating system that is installed in a virtual machine is used to configure SDN. The SDN network, which presently consists of a host and controller, creates a flat tree topology. The switches that will be utilized are ones that support OpenFlow and are linked to the RYU controller.

A.Mininet

Mininet is used to design the topology. According to researchers' discussions in the related work area [12], mininet offers superior features for topology simulation. sFlow and mininet are connected because sFlow provide statistics on network data [13]. We can set up many hosts and switches with mininet so they can communicate with each other via RYU controller.

B. sFlow-RT

sFlow-RT uses an add-on mininet dashboard so they may talk to each other and share network flow statistics [14]. Since the mininet is the most popular platform for creating SDN topologies, it is deployed. It is possible to create the necessary topology in a mininet with several hosts and controllers. Three switches and four hosts are now being evaluated for a flat tree architecture in this scenario. These switches and the number of hosts can be increased in accordance with new requirements because mininets are scalable and simple to set up. Three switches with OpenFlow enabled and four hosts make up the topology configuration seen in Fig. 3. sFlow-RT The mininet dashboard add-on and the sFlow-RT setup are complete. Integration of mininet and sFlow so that sFlow-RT may inspect and monitor topology produced in mininet. sFlow-RT offers graphical user interface port statistics that facilitate the analysis of flow traffic inside the mininet topology depicted in Fig. 3. An open-source platform called sFlow is used to monitor traffic using statistical and real-time traffic sampling techniques. Its GUI for viewing network performance and traffic patterns is improved. The Fig. 3 test bed will require sFlow because sFlow will be useful for DDoS attack detection later in the project.

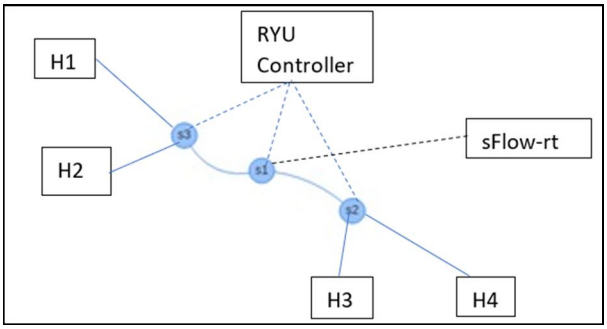

Figure 3. Topology of the proposed scenario in sFlow

C. RYU Controller

The most popular controller used by researchers to create SDN networks is the RYU controller [15]. This test bed uses the RYU controller, an open-source SDN controller. OpenFlow switches and numerous more protocols are supported by the controller script, which is written in Python. Additionally, it facilitates compatibility when building well specified APIs. Since RYU controller is compatible with OpenFlow switches, it has been evaluated and approved by NEC, IBM, and HPC.
Additionally, it is compatible with OpenFlow protocols up to version 1.5 [2]. RYU controller supervises the switches that are linked to it by keeping track of all traffic, both new and old, in the network.

D. Wireshark

Wireshark is used to capture the communication between controller and switches [16]. Open Flow Protocol captures the communication between Open vSwitch and controller before starting the actual communication. So the control plane communication is seen Wireshark which shows Open Flow version 1.3 packets been captured for the communication.

E. Performing DDoS Attack

DDoS attacks have been studied by researchers in paper [17]. Two types of DDoS attack can be implemented on the topology discussed. DDoS using UDP flood messages and using TCP SYN flood messages is performed.

V. DDoS Attack detection and results

DDoS attack is performed using two methods as discussed in the previous section. So, a normal ping form host 1 to host 2 is captured in sFlow and parameters are noted down. A normal operation of the topology can be seen in Fig. 4 below. It shows the normal traffic flow and the number of bits per second exchanged. Number of sFlow

agents used in this case it is one, sFlow bytes currently handled by the sFlow and sFlow packets per second. Hence sFlow shows the transaction happening during the H1 ping H2 via the OpenFlow protocol. Normal sFlow packets are filtered and read in the Wireshark. Wireshark helps to understand and troubleshoot if any issues related to the communication between the host, Open vSwitch and the sFlow detection. Similarly normal flows are detected in the RYU manager. If the packets are not matched on highest priority, then they are forwarded to lowest priority. Once the connection between H1 and H2 is established next ping onwards the packet will not move to controller since it has learnt and forwarded the flows to the switches. Therefore, when we see the dumps for the next ping the number of packets in the RYU controller it is constant 102 in this case and the number of bytes is 9817 bytes. This section shows the normal behaviour of the topology and its interpretation using sFlow, Wireshark and RYU controller output. Fig. 5 depicts the packet flow in the normal scenario when H1 pings H2.

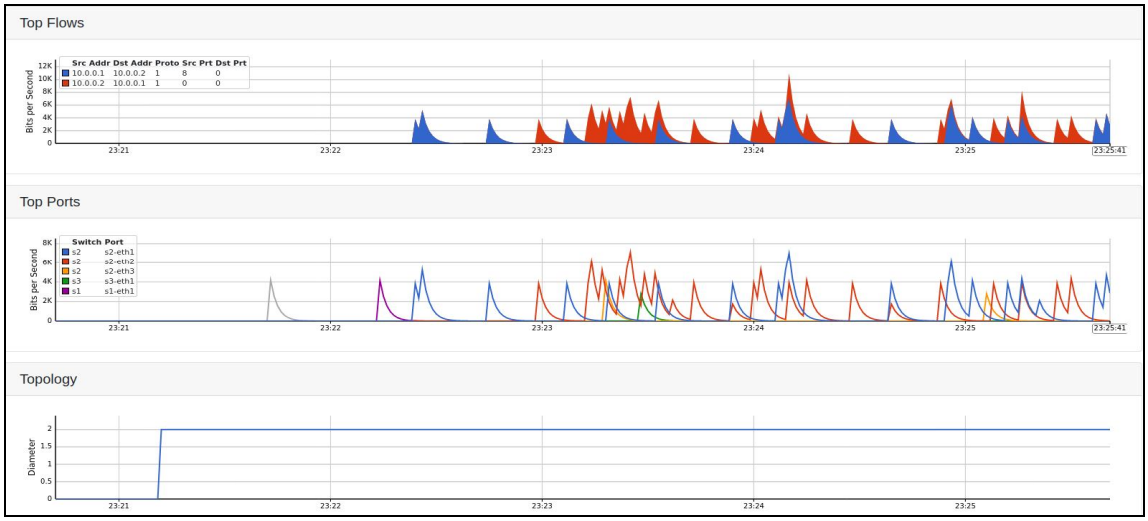

Figure 4. Normal Traffic Flow before the attack

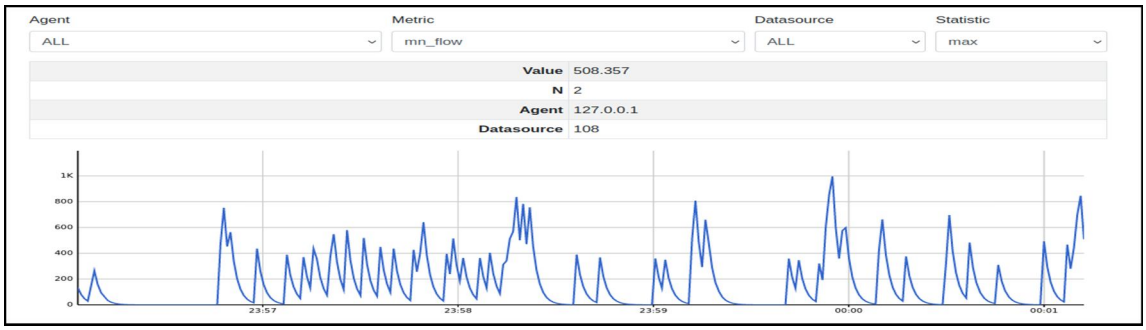

Figure 5. Mininit Packet flow before attack

The attacks are performed using mininet and the topology is same linear topology with 3 Open vSwitches and 4 hosts. In Fig.6 flood attack is detected in the sFlow-RT under Top Flows. The number of bits is increased rapidly as soon as the attack is initiated. Number of bits per second is increased to 10 Megabits per second which is increased from 6 kilobits per second. Both the values are considered highest value of number of bits processed in normal as well as UDP attack case. The switch ports are busy handling the bits when attack is initiated. This increases the traffic handling of the switch, and this traffic is not real traffic, so it is consuming the useful bandwidth of the switch. If this scenario continues the switch buffers will overflow.

Various parameters are used in the GUI to detect the DDoS attack. sFlow bytes is increased from 2.13Kbps to 7.03Mbps. sFlow packet throughput is increased from 1.19 pps to 653 pps. This increases the processing of the CPU processor. CPU processes is increased from 0.66% to 18.2% and CPU System from 18.3% to 97.7%. The consequence of this is processing of the other requests is hampered since CPU processes get busy handling the attack packets. The number of packets handled by the RYU controller is increased from 102 to 23726. Number of bytes handled by the RYU controller before attack was 9817 bytes which is increased to 1001068 bytes after

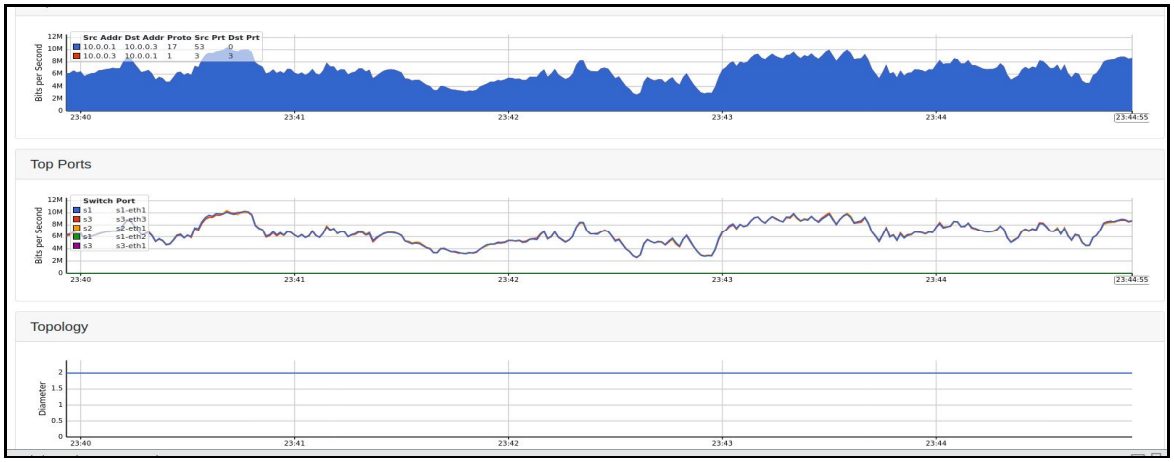

Figure 6. UDP Flood detection in sFlow-RT

the UDP attack. Wireshark detects the abnormal flow of packets in sFlow and highlights the same in the packet capture window.

A TCP SYN attack is also done on the considered topology and the detection of TCP SYN attack is done in the similar way as in the case of UDP attack detection. The increase in the number of bits per second processing in shown in Fig.7 below

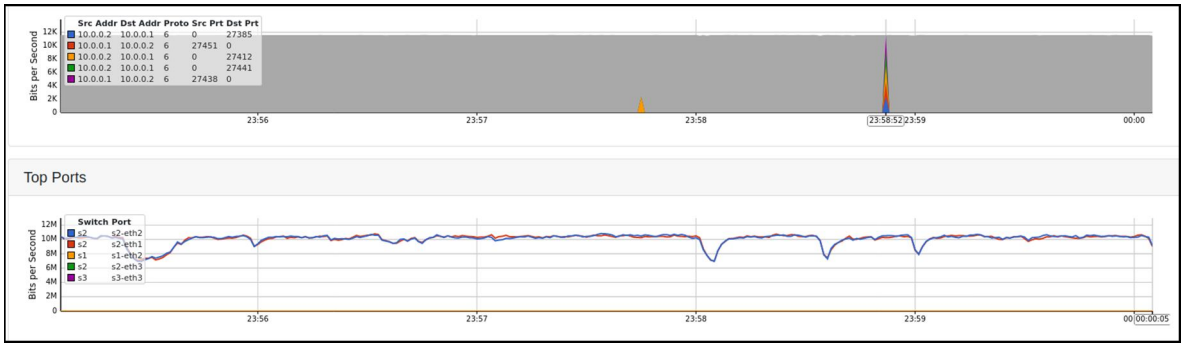

Figure 7. TCP SYN detection in sFlow-RT

The findings of the DDoS attack performed on the linear topology is tabulated in Table I below. Table is depicting two scenarios one is when there is normal movement of packet flow as in case when H1 pings H2. When attack is performed on the topology all the parameters like number of packets being processed by RYU controller, Open vSwitch, CPU processes and CPU system utilization is shown in Table I. The attack reduces the CPU process efficiency by putting a load on CPU processes.

TABLE I. PARAMETERS MEASURE FOR UDP FLOOD ATTACK

Parameters	Normal ping between two hosts	After attack is initialized	Percentage change
Number of packets processed	102	23726	23260.78
Throughput (number of bytes)	9817	1001068	10197.29
Throughput	6kbps	10Mbps	1666.67
sFlow bytes	2.13kbps	7.03Mbps	3301.40
sFlow packet throughput	1.19 pps	653pps	54873.94
CPU processes	0.66%	18.2%	17.54
CPU System	18.3%	97.7%	79.4

When TCP SYN attack is initialized on the topology its effect on the parameters of the switch is tabulated in Table II below. Number of packets processed by switch is increased from 102 to 20117888 after the attack is initiated. The attack is initialized on H1 towards the other host H2 via the Open vSwitch. The packets processed by Open vSwitch is increased from 9817 to 1086365232. Parameters are also measured using sFlow which shows increase in the processing of bits per second from 6 kbps to 12 Mbps. Packets processed by sFlow is

increased from 1.19 packets per second to 355 pps. CPU process increases from 0.66% to 11%. CPU system utilization increases from 18.3% to 59.6%. TCP SYN attack does not allow to complete the three-way handshake process and because of this whenever a TCP SYN request is sent it is automatically reset which is indicated by RST flag in TCP header. So, because of this control plane is consuming the bandwidth which was supposed to be used for data. This stops TCP from creating a connection establishment between tw hosts.
Both UDP and TCP attack impact packet processing capacity of the Open vSwitch as discussed in Table I and Table II. CPU processes, packet throughput and CPU System utilization increases drastically. A solution is proposed to this using Blockchain designing the flows in RYU controller such that it detects the attack within few milliseconds. This will mitigate the attack at the earliest time. sFlow can be used to inform the RYU controller of the possible start of the attack with increase in the flow of packets in the switch. This data can then be used by the RYU controller to mitigate the attack. So once attack is mitigated all the parameters are restored to normal values. Also, it will take care of any possible attack in future by using port scanning mechanism and using output form sFlow.

TABLE II. PARAMETERS MEASURE FOR TCP SYN FLOOD ATTACK

Parameters	Normal ping between 2 hosts	After attack is initialized	Percentage change
Number of packets processed	102	20117888	196194007.8
Throughput (number of bytes)	9817	1086365232	110661.631
Throughput	6kbps	12Mbps	2000
sFlow bytes	2.13kbps	5.36Mbps	2516.43
sFlow packet throughput	1.19 pps	355pps	298.31
CPU processes	0.66%	11%	10.34
CPU System	18.3%	59.6%	41.3

VI. CONCLUSIONS

SDN is very much useful in managing a big network such service provider network. There are lot of threats that can affect regular functionality of the Open vSwitches used in communication. One of these threats is the DDoS attack which can take place in various forms. Two varieties of DDoS attacks are considered in this paper and its overall effect on parameters such as throughput, packet processing and CPU utilization is discussed. A solution is proposed that assists in mitigating DDoS attack at earlier stage using applications viz. sFlow and Block chain thereby restoring the system to its normal operation. After mitigating the attack and restoring system to its normalcy continuous monitoring is possible to avoid further attack.

REFERENCES

[1] D.B. Rawat, and S. Reddy, “Software defined networking architecture, security and energy efficiency: A survey.,” *IEEE Communications Surveys & Tutorials 19*, no. 1 (2016): 325-346.

[2] Karmakar, K. Krishna, V. Varadharajan, and U. Tupakula. "Mitigating attacks in software defined networks." Cluster Computing 22 (2019): 1143-1157

[3] E. Fayez, and R. Pietro, "DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges." *Future Generation Computer Systems*, no.122 (2021): 149-171

[4] Hu, Fei, Q. Hao, and K. Bao. "A survey on software-defined network and openflow: From concept to implementation." *IEEE Communications Surveys & Tutorials 16*, no. 4 (2014): 2181-2206

[5] Kreutz, Diego, F. Ramos, P. Verissimo, C. Rothenberg, S. Azodolmolky, and S Uhlig. "Software-defined networking: A comprehensive survey." *Proceedings of the IEEE 103*, no. 1 (2014): 14-76

[6] R. Swami, M. Dave and V. Ranga, "Addressing Spoofed DDoS Attacks in Software-defined Networking," *6th International Conference for Convergence in Technology (I2CT)* , Maharashtra, India, 2021, pp. 1-7, doi: 10.1109/I2CT51068.2021.9418202.

[7] S. Asadollahi, B. Goswami (2018) “RYU controller’s scalability experiment on software defined networks” In: 2018 *IEEE international conference on current trends in advanced computing (ICCTAC)*

[8] De Paolis LT, De Luca, R Paiano (2018) “Sensor data collection and analytics with Things-Board and Spark Streaming”. In: 2018 *IEEE workshop on environmental, energy, and structural monitoring systems (EESMS),*2018

[9] Agarwal, Sashank. "Real-time DDoS Detection and Mitigation in Software Defined Networks using Machine Learning Techniques." (2022)

[10] B. Pande, G. Bhagat, S Priya (2018) “Detection and mitigation of DDoS in SDN. “In: Eleventh International conference on *contemporary computing (IC3)*, 2–4 Aug 2018 R. Nicole

[11] SS Bhunia, M. Gurusamy (2017) “Dynamic attack detection and mitigation in IoT using SDN” In: 27th *International Telecommunication Networks and Applications Conference (ITNAC). IEEE*

[12] Gupta, Neelam, M. Maashi, S. Tanwar, S. Badotra, M. Aljebreen, and S. Bharany. "A comparative study of software defined networking controllers using mininet." *Electronics 11*, no. 17 (2022): 2715.
[13] Dholakiya, Dhruvkumar, T. Kshirsagar, and A. Nayak. "Survey of mininet challenges, opportunities, and application in software-defined network (sdn)." *Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2020*, Volume 2 (2021): 213-221.
[14] Leal, Alexander, J. Botero, and E. Jacob. "Improving early attack detection in networks with sFlow and SDN." In *Applied Computer Sciences in Engineering: 5th Workshop on Engineering Applications, WEA 2018, Medellín, Colombia, October 17-19, 2018, Proceedings, Part II 5*, pp. 323-335. Springer International Publishing, 2018.
[15] Albu-Salih, A. Taima. "Performance evaluation of RYU controller in software defined networks." *Journal of al-qadisiyah for computer science and mathematics* 14, no. 1 (2022): Page-1.
[16] Pavithirakini, D. D. M. M. Bandara, C. N. Gunawardhana, K. K. S. Perera, B. G. M. M. Abeyrathne, and Dhishan Dhammearatchi. "Improve the Capabilities of Wireshark as a tool for Intrusion Detection in DOS Attacks." *International Journal of Scientific and Research Publications* 6, no. 4 (2016): 378-384.
[17] Singh, Chandrapal, and Ankit Kumar Jain. "A Comprehensive Survey on DDoS Attacks Detection & Mitigation in SDN-IoT Network." *e-Prime-Advances in Electrical Engineering, Electronics and Energy* (2024): 100543.