International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 06 | Jun 2022 www.irjet.net p-ISSN: 2395-0072

Denial of Service Attacks in Software Defined Networking - A Survey

Sreejesh N. G.[1], Sabina M. A.[2]

[1]P G Scholar, Department of Computer Science and Engineering,
Rajiv Gandhi Institute of Technology, Kottayam, Kerala, India
[2]Assistant Professor, Department of Computer Science and Engineering,
Rajiv Gandhi Institute of Technology, Kottayam, Kerala, India

---

Abstract - *The word Software Defined Networks (SDN) came into limelight only a few years ago. But it has become the backbone for various popular network technologies such as data centers, cloud storage, mobile communication, Internet of Things and even in small business environments, thanks to the flexibility, programmability, scalability and centralized coordination that brings to the entire network. Even though the introduction of SDN brought easiness in network management, it is susceptible to all the attacks that may happen in the traditional network. One such common attack is saturation attack, means saturating the network resources using malformed packets. In happens in two forms; Denial of Service attacks and Distributed Denial of Service attacks. Many proposals have been brought since the evolution of SDN to detect and mitigate these attacks. This survey brings an overview of some of such proposals, their effectiveness and possible comparisons.*

Key Words: Software Defined Networking, saturation attacks, DoS, DDoS, IoT, control plane, data plane, OpenFlow, northbound and southbound interfaces.

1. INTRODUCTION

The evolution of network has gone through different phases before reaching the fully connected world. The traditional networking scenario uses routers as the backbone of the network paradigm. Apart from it, gateways and switches make their specific roles wisely. Transfer a packet of data from a source to destination needs proper end to end connectivity and through this connection, physical or logical, packets are travelled. This transfer needs proper routing and forwarding. The traditional network does this task in a fantastic manner with the help of different protocols and addressing schemes. The problem arises when some functionalities of the intermediate network need to be changed. This brings a tedious thing because; most of the functionalities are fixed into the network devices. Changing the functionality require changing the whole or part of the network devices. This causes headache both practically and economically. Here comes the role of Software Defined Networking (SDN).

A network device can have a data plane to forward the incoming packets, a control plane to manage and control the internal processing of the device, such as creating the routing table, executing specific algorithms etc, and an application plane to access the device either for monitoring the device, managing the control plane by the administrators or executing business specific applications. In the traditional network components such as routers and switches, the control plane and the data plane are in a combined form. If the network does not need any change, this form is good method. But as the network technology advances more functionality and changes need to be incorporated. Hence the control plane must be separated from the data plane, as it is the control plane which needs reformation every time.

2. BACKGROUND

The SDN technology separates the control plane from the data plane [1][10][14]. The data plane devices are simply forwarding devices. The control plane is mostly a powerful system to which the data plane devices are connected directly or indirectly. This system is called the SDN controller. The control plane functionalities are programmed into the SDN controller. This is why the SDN is aid to be a programmable network. One major duty of the controller is to instruct the forwarding device what to do with the incoming packets. Since the most common action is forwarding the packets, this device can be considered as a special type of switch. The application plane (otherwise called management plane) mostly lies on the controller itself, or it may be in a system directly accessible to the controller. The applications are used to monitor the network and its various parameters, business specific functionalities or as administrative access path to the controller. The architecture of SDN is shown in Fig. 1 [14].

The interface between the data plane and the control plane is called the southbound interface. The forwarding devices communicate with the controller through this interface. The packets that flow through this interface are generally packet-in messages, packet-out messages and flow modification messages. The interaction need to follow some protocols for effective communication. The protocol used in the southbound interface is the OpenFlow protocol [2][3]. The OpenFlow follows an event based way of communication. The switches that follow the OpenFlow protocol are called OpenFlow Switches, which forms the data plane of the SDN architecture.

The interface between the control plane and the application plane is called the northbound interface. The protocol in this interface varies with the APIs used in the interface, and hence there is no specific protocol in this interface.

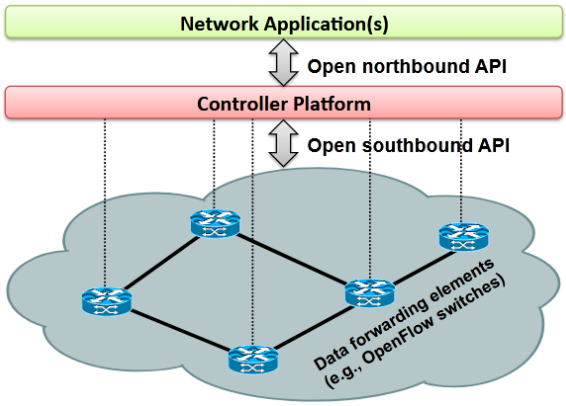

Fig -1: SDN Architecture

The features that highlights the SDN from traditional network comprises of its ability to program the network without changing any of the forwarding device. Any policies or functionalities that need to be added or changed can be done in the controller through programs, which will take effect in the whole network. This will save huge effort in changing the network infrastructure to implement the change. Another feature is the fine grained access control bases on the policies assigned by the SDN controller. This is highly useful in IoT based environments as they do not have in-built access control mechanisms in their light weight devices [11][12].

2.1 Working of SDN

The core component in the OpenFlow switch is the flow table. It is a huge table containing two major fields; match field and corresponding action field. The match table can contain parameter that can be checked with the incoming packets. If any match occurs, the corresponding action is taken. Some common actions are; forwarding the packet to a specified port, drop the packet, etc. The flows will have priorities so that the matches with higher priorities are checked first. If no match is there, it is called a table-miss. Then the default action is taken, which is to forward the packet to the controller, encapsulating it in a message called packet-in message. On getting he packet-in message, the controller unwraps it and decides what is to be done with that packet. The packet is then sent back to the switch putting it into a packet-out message, along with an action for this packet. The controller may then send a flow rule to that switch specifying the match and action for the similar packets that may come in the future so that when the packet comes later to the switch, it will create a table-hit and the packet need not be sent to the controller, saving the bandwidth in between.

The SDN controller runs code for handling the packet-in messages and other event-messages. Examples of such controller code/softwares are RYU [4], POX [5], NOX [6], Floodlight [7] etc. Most of the controller softwares are open in nature so that developers are free to modify it or add more functionalities as desired.

2.2 Attacks in SDN

The major feature and at the same time the most prominent vulnerability of the SDN is its centralized control. This weakness is used by the attackers to act upon the controller to perish the entire network. Most attacks in traditional network is applicable to SDN too [8][9][10]. Saturation attack is the major SDN aimed attack. Here the major areas of SDN aimed attacks are the switch, the link between the controller and the switch, and the controller itself. Exhausting the resources in all the three will result in denial of services to the hosts connected to the network. Hence it is a type of Denial of Service (DoS) attack. It is possible for attackers to create a botnet by including more zombie systems into the attack scenario and hence this can lead to Distributed DoS (DDoS) attack.

In a (D)DoS attack, the attacker creates spoofed packets and flood them into the switch. These will be queued up at the switch causing a lot of table miss and thereby lot of packet-in message to the controller. Hence, the switch resources will get exhausted and the link between the switch and the controller filled with useless packet-in messages. The controller on the other hand will unwrap each of this, try to find action for it which will go in vain. It will tell the switch to flood these packets and create useless entries in the flow table. Thus the controller CPU time and memory also get wasted. The packets who suffer are the benign packets which cannot reach the switch or the controller and the service for its user is denied.

3. ATTACK DETECTION AND MITIGATION

As DoS attacks can happen in any networks, so is in SDN. Here the difference is that the attacks can be host-aimed or SDN-aimed. The host-aimed attacks are possible in all networks in dependent of whether it is legacy [14] or SDN based [8][9][10].

TCP connections are heavier than other connection less protocols as there are different handshakes and table managements as far as a normal host is considered. Most TCP attacks use SYN flags to exhaust the TCP connection tables of the target. To solve this type of attack, the controller can act as a proxy by managing a table of active TCP states as seen in Avant-Guard [16]. Thus the flows can be classified into those which complete the handshake and those which do not.

Based on this, it can create flow rules to manage the flows which do not complete the TCP handshakes.

Apart from TCP, other protocols can also create attacks especially UDP and ICMP. All the attacks will cause a disaster to the networks and need to be detected with the combined action of the switch and the controller. NEOD describes this scenario and a remedy [17]. But this must not increase the CPU load of the controller and the switch.

Sometimes the network operating system (NOS) itself can be created with primary focus to prevent attacks that may happen to the controller by compartmentalizing the network applications so that the services would work in side sandboxes and does not cause much problems to the network. Rosemary [18] is designed with such a view. Hence it is possible to say that the OS is application-centric rather than flow-centric.

One of the common methods is to first detect the presence of any attack and mitigate it by means of different algorithms. This detection and mitigation strategy is used by many researchers in their own methods such as FloodGuard [19], SGuard [20], Flokeeper [22], SDNGuard [24] and FloodDefender [25] . Whenever an attack starts, the switches will be getting overwhelmed. To overcome such situation, the general ideas are either to divert the packets to somewhere else, say neighbor switches [25] without losing the inport information, or to store the packets temporarily in a cache for later processing [19][20][24]. But using the additional cache for packet storage is practically very difficult. Also it needs extra cost to add the additional hardware cache. Classifying the packets into benign and illegal is another tedious thing. Different classifiers are used for this such as SVM [25], Self Organizing Maps [20], etc. In small networks, this will not take much time, but when the network grow bigger in size, the real time classification must not take much time as it may cause severe packet transfer delay. Another method of finding the attack is the anomaly detection in the network [23]. Maddu *et. al.* uses a probabilistic model for such classification [24].

When the number of spoofed packets increases at switch, this will increase the number of packet-in messages to the controller. Thus the switch-controller link will get busier and this this may lead to link saturation. Hence by monitoring the link bandwidth, it is possible to detect the presence of flooding at the switch and there by predict the attack [21]. The prediction will not be crisp as the network behaviour can be changed at any time. Thus only a fuzzy model is feasible in predicting the future bandwidth utilization and thereby the chance of attack.

Most of the alleviation methods are focused on the controller. Recently experiments are being happened to provide some sort of intelligence to the openflow switches too, so that they can do more than just forwarding the packet based on the flow table. Such an initiation is the Data Plane Development Kit (DPDK) that consists of libraries installed in the switch to accelerate packet processing and for efficient computing. Based on the DPDK the attack detection can be made faster by processing each packet in a very speedy way. Newer methods such as DPDK based DDoS Detection frameworks [26] are based on such packet processing. With this the attack detection happens at the data plane itself and mitigation at the control plane.

One of the difficult attacks to mitigate is the Low Rate attack, which occurs as spikes at random interval of time. To identify such attacks different methods need to be combined and collaborated. Tang *et. al.* has introduced a model for classification based on performance and features of the traffic [27]. It combines both machine learning and flow frequency analysis.

A comparison of some of the previous proposals are shown in table 1.

Table -1: Comparison of Different Methods

Proposal	Protocols	Additional Hardware	Controller
AvantGuard [16]	TCP	No	POX
FloodGuard [19]	All	Yes	POX
SGuard [20]	All	Yes	NOX
BWManager [21]	All	No	Floodlight
DAISY [23]	All	No	RYU
SDNGuard [24]	All	Yes	POX
FloodDefender [25]	All	No	RYU

3.1 Security Issues in SDN IoT

The SDN can be introduced as the backbone of the IoT network, so that the controller will get an entire view on the whole IoT topology [28][29]. The sensor data packets can be inspected by the controller and know the complete attributes of the packets. This enables the controller to manage the IoT network in a finer level.

IoT architecture also is not escaped from DoS attacks. With spoofed flood and junk value packets, the sink devices may get overwhelmed and the valuable sensor packets cannot reach the sink, causing DoS situation. As SDN can

handle the devices and packet flow in a granular way, different proposals have been introduced by implementing SDN into IoT environment. Some are just to fine tune the data flow and some lightly approaches the security aspects also. Common attacks that may happen to an IoT network include malware injection, DDoS attacks, spoofing/masquerading and man-in-the-middle attack [12]. As in data centers and clouds, DoS and DDoS attacks in the IoT environment may lead to failure of collecting sensor data and there by hindering the intended services to be run at the proper time. The attacks may happen from inside or outside the IoT network. This can be controlled by limiting the access to authenticated devices only and controlling the access of network services in a pre-defined manner. This is possible using SDN.

Z. Qin *et. al.* [31] proposed MINA which uses the programmability and flow control feature of the SDN. It focuses on Network Services and scheduling flows. The network services are provided as programmable features and SDN's inherent flow table mechanism is used for scheduling flows. This resists the illegal flows in the network and thereby reducing the chance of DoS attacks. But, there is no mentioning about which devices has to access the network and which services are allowed for them. IoTPot and IoTBox [30] are some sort of honey pot and sand boxing methods to trap the attacker and there by stopping it from generating a DoS attack. An entropy based method in [32] finds anomaly in the packet flow by constantly monitoring the data flow and thereby detects and mitigates the illegal packets. On larger IoT networks of heterogeneous devices, clusters of devices can be formed, which are managed by one SDN controller for each cluster and all the cluster controllers can be coordinated by a master controller [33]. Here the cluster controllers will classify the abnormal flows using anomaly detection through Support Vector Machine (SVM) and will report to the master controller. The master will decide the actions/flow rules to manage and control the flows in the network. Krishnan P *et. al.* [34] proposed a loosely coupled integration scheme and tightly coupled integration scheme. In the former, the controller apps take care of the attacks whereas the latter does the detection through the switch and its security controller.

3.2 Implementation of Proposals

To implement and test the proposals, setting up a hardware environment is very costly. Hence the researchers use an emulator tool called "Mininet" [35][36][37] for creating the wired and wireless setup. It is highly scalable and powerful so that it can create virtual hosts and Open Virtual Switches. The hosts are provided with terminals to access its functionality and the switches are as powerful as the real ones. With specific tools such as Scapy [38][39][40] and hping [41], benign and spoofed traffic can be created inside the network. Performance of the systems such as bandwidth and throughput can be measured using iperf [42] like utilities. The controller can be from a list of choices such as RYU, POX, NOX, Floodlight etc.

4. CONCLUSIONS

This paper provides a brief overview of DoS attacks in SDN and some mitigation measures proposed for various contexts such as data warehouses and IoT environments. As the controller is the key element in SDN, any attempt which causes hindrance to the controller or the link to the controller will decrease the functionality of the network, sometimes destroying the entire communication. It can be seen that the measures against DoS attacks are well suited for DDoS attacks too. The measures reveal that the attack mitigation is possible through speculative and efficient management of resources and prompt and accurate classification of illegal packets from benign ones. Furthermore, this paper brings the methodology used in the preventive measures, the features they used and the limitations they bear. Each new method tries to overcome the gap found in the previous ones. Overall, a researcher can use this summary to well prepare for finding new methods to detect and mitigate DoS and DDoS attacks in more effective ways.

REFERENCES

[1] Kirkpatrick, Keith. "Software-defined networking." Communications of the ACM 56.9 (2013): 16-19. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[2] N. McKeown et al., "OpenFlow: Enabling innovation in campus net-works," ACM SIG-COMM Comput. Commun. Rev., vol. 38, no. 2,pp.69–74, Mar. 2008.

[3] OpenFlow Switch Specification [online] https://opennetworking.org/wpcontent/uploads/2014/10/openflow-switch-v1.3.5.pdf

[4] RYU SDN Framework [online] https://ryu-sdn.org/

[5] Kaur, Sukhveer, Japinder Singh, and Navtej Singh Ghumman. "Network programmability using POX controller." ICCCS International conference on communication, computing & systems, IEEE. Vol. 138. sn, 2014.

[6] Gude, Natasha, et al. "NOX: towards an operating system for networks." ACM SIGCOMM computer communication review 38.3 (2008): 105-110.

[7] Floodlight Controller [online] https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview

[8] Chica, Juan Camilo Correa, Jenny Cuatindioy Imbachi, and Juan Felipe Botero Vega. "Security in SDN: A

comprehensive survey." Journal of Network and Computer Applications 159 (2020): 102595.

[9] Ahmad, Ijaz, et al. "Security in software defined networks: A survey." IEEE Communications Surveys & Tutorials 17.4 (2015): 2317-2346.

[10] Benzekki, Kamal, Abdeslam El Fergougui, and Abdelbaki ElbelrhitiElalaoui. "Software-defined networking (SDN): a survey." Security and communication networks 9.18 (2016): 5803-5833.

[11] Flauzac, Olivier, et al. "SDN based architecture for IoT and improvement of the security." 2015 IEEE 29th international conference on advanced information networking and applications workshops. IEEE, 2015.

[12] Karmakar, Kallol Krishna, et al. "SDN-enabled secure IoT architecture." IEEE Internet of Things Journal 8.8 (2020): 6549-6564.

[13] Das,, Resul, Abubakar Karabade, and Gurkan Tuna. "Common network attack types and defense mechanisms." 2015 23nd signal processing and communications applications conference (siu). IEEE, 2015.

[14] Kreutz, Diego, et al. "Software-defined networking: A comprehensive survey." Proceedings of the IEEE 103.1 (2014): 14-76.

[15] Abdullaziz, Osamah Ibrahiem, Li-Chun Wang, and Yu-Jia Chen. "HiAuth: Hidden authentication for protecting software defined networks." IEEE Transactions on Network and Service Management 16.2 (2019): 618-631.

[16] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS), 2013, pp. 413–424.

[17] S. Song, S. Hong, X. Guan, B.-Y. Choi, and C. Choi, "NEOD: Network embedded on-line disaster management framework for software defined networking," in Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM), May 2013, pp. 492–498.

[18] Shin, Seungwon, Yongjoo Song, Taekyung Lee, Sangho Lee, Jaewoong Chung, Phillip Porras, Vinod Yegneswaran, Jiseong Noh, and Brent Byunghoon Kang. "Rosemary: A robust, secure, and high-performance network operating system." In Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, pp. 78-89. 2014.

[19] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS attack prevention extension in software-defined networks," in Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw., Jun. 2015, pp. 239–250.

[20] Wang, Tao, and Hongchang Chen. "SGuard: A lightweight SDN safeguard architecture for DoS attacks." China Communications 14, no. 6 (2017): 113-125.

[21] Wang, Tao, Zehua Guo, Hongchang Chen, and Wei Liu. "BWManager: Mitigating denial of service attacks in software-defined networks through bandwidth prediction." IEEE Transactions on Network and Service Management 15, no. 4 (2018): 1235-1248.

[22] S. Gao, Z. Li, B. Xiao, and G. Wei, "Security threats in the data plane of software-defined networks," IEEE Netw., vol. 32, no. 4, pp. 108–113, Jul. 2018.

[23] Imran, Muhammad, Muhammad Hanif Durad, Farrukh Aslam Khan, and Haider Abbas. "DAISY: A detection and mitigation system against denial-of-service attacks in software-defined networks." IEEE Systems Journal 14, no. 2 (2019): 1933-1944.

[24] Maddu, Jeevan Surya, Somanath Tripathy, and Sanjeet Kumar Nayak. "SDNGuard: An Extension in Software Defined Network to Defend DoS Attack." In 2019 IEEE Region 10 Symposium (TENSYMP), pp. 44-49. IEEE, 2019.

[25] Gao, Shang, Zhe Peng, Bin Xiao, Aiqun Hu, Yubo Song, and Kui Ren. "Detection and Mitigation of DoS Attacks in Software Defined Networks." IEEE/ACM Transactions on Networking 28, no. 3 (2020): 1419-1433.

[26] Varghese, Josy Elsa, and Balachandra Muniyal. "An Efficient IDS Framework for DDoS Attacks in SDN Environment." IEEE Access 9 (2021): 69680-69699.

[27] Tang, Dan, et al. "Performance and Features: Mitigating the Low-Rate TCP-Targeted DoS Attack via SDN." IEEE Journal on Selected Areas in Communications 40.1 (2021): 428-444.

[28] Tayyaba, Sahrish Khan, et al. "Software defined network (sdn) based internet of things (iot) a road ahead." Proceedings of the international conference on future networks and distributed systems. 2017.

[29] Flauzac, Olivier, et al. "SDN based architecture for IoT and improvement of the security." 2015 IEEE 29th international conference on advanced information networking and applications workshops. IEEE, 2015.

[30] Pa, Yin Minn Pa, et al. "IoTPOT: Analysing the Rise of IoT Compromises." 9th USENIX Workshop on Offensive Technologies (WOOT 15). 2015.

[31] Qin, Zhijing, et al. "A software defined networking architecture for the internet-of-things." 2014 IEEE network operations and management symposium (NOMS). IEEE, 2014.

[32] Galeano-Brajones, Jesus, et al. "Detection and mitigation of dos and ddos attacks in iot-based stateful sdn: An experimental approach." Sensors 20.3 (2020): 816.

[33] Bhunia, Suman Sankar, and Mohan Gurusamy. "Dynamic attack detection and mitigation in IoT using SDN." 2017 27th International telecommunication networks and applications conference (ITNAC). IEEE, 2017.

[34] Krishnan, Prabhakar, Jisha S. Najeem, and Krishnashree Achuthan. "SDN framework for securing IoT networks." International Conference on Ubiquitous Communications and Network Computing. Springer, Cham, 2017.

[35] Mininet Walkthrough [online] http://mininet.org/walkthrough/

[36] De Oliveira, Rogerio Leao Santos, et al. "Using mininet for emulation and prototyping software-defined networks." 2014 IEEE Colombian conference on communications and computing (COLCOM). Ieee, 2014.

[37] Fontes, Ramon R., et al. "Mininet-WiFi: Emulating software-defined wireless networks." 2015 11th International Conference on Network and Service Management (CNSM). IEEE, 2015.

[38] Rohith, R., Minal Moharir, and G. Shobha. "SCAPY-A powerful interactive packet manipulation program." 2018 international conference on networking, embedded and wireless systems (ICNEWS). IEEE, 2018.

[39] Biondi, Philippe. "Scapy: explore the net with new eyes." Technical report, Technical report, EADS Corporate Research Center (2005).

[40] Scapy [online] https://scapy.net/

[41] hping3 [online] https://linux.die.net/man/8/hping3

[42] iperf [online] https://iperf.fr/iperf-doc.php

BIOGRAPHIES

Sreejesh N. G. is a P. G. Scholar of Computer Science and Engineering at Rajiv Gandhi Institute of Technology, Kottayam, Kerala, India. He has completed his B.Tech in Computer Science and Engineering. His areas of interest include Computer Networks, SDN and Network Security.

Sabina M. A. is an Assistant Professor in the Department of Computer Science and Engineering at Rajiv Gandhi Institute of Technology, Kottayam, Kerala, India. She has completed her M.Tech in Computer Science and Engineering. She has more than 5 years of experience as Lecturer and her areas of interest include Computer Networks and Machine Learning.