International Journal of Pure and Applied Mathematics
Volume 119 No. 10 2018, 691-698
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

ijpam.eu

Effective Threshold Defence against DOS Attack on SDN Controller

[1]Akhil Raj, [2]Anjali S Bhat and [3]Leena Vishnu Namboothiri
[1]Department of Computer Science and IT,
Amrita School of Arts and Sciences, Kochi,
Amrita VishwaVidyapeetham, India.
akhilraj949@gmail.com
[2]Department of Computer Science and IT,
Amrita School of Arts and Sciences, Kochi,
Amrita VishwaVidyapeetham, India.
asb.anjali@gmail.com
[3]Department of Computer Science and IT,
Amrita School of Arts and Sciences, Kochi,
Amrita VishwaVidyapeetham, India.
vleena@gmail.com

Abstract

With the increase in digitization, cyber attacks have been on the rise. In this paper, we discuss the adverse effects a DOS attack has on an SDN architecture. Due to this centralized architecture, the control logic may be prone to a variety of security threats, vulnerabilities and other attacks. With the help of a hashing mechanism that is triggered on the basis of threshold values, we diagnose and prevent a DOS attack from disrupting the entire network. This paper illustrates efficient and cost effective algorithms that can be implemented in an SDN network to safeguard it from the ruinous effects of a DOS attack.

Key Words:SDN(Software Defined Networking), Open flow, DOS(Denial-of-Service), Hashes ,TTL(Time-to-Live),Threshold.

1. Introduction

Business strategies have been evolving vastly over time with the emergence of cloud computing, leading consumers to shift from traditional network models to cloud computing models. This developed a need for high bandwidth due to the self-provisioning nature of cloud computing which demanded a highly scalable, dynamic and cost effective network. This led to the development of SDN, which has created revolutionary impacts and gained immense popularity due to its highly decoupled design. SDN was aimed at improving the network performance and monitoring capabilities in the network. It addresses the issue of high bandwidth requirement by providing a more flexible and programmable network[1].

SDN Architecture

SDN is a dynamic, virtualized network that follows a layered architecture where the control plane is physically separated from the forwarding plane. The control plane forms the central logic or the "brain" of the network that interacts with and controls both the APIs: the northbound interface and the southbound interface as depicted in ***Figure 1***. The northbound interface basically contains user applications that collects/requests resources from/to the controller whereas the southbound consists of hardware devices that perform forwarding functions. In contrast to the traditional network architecture which was based on a simple client-server model, SDN provides a programmable network[5] with a control plane enabling the administrators or the operators of the network to easily configure, administer, optimize and orchestrate[6] network resources efficiently. Though several other communication protocols such as BGP[7], GMPLS[7], PCEP[7] etc. exist, SDN controllers are commonly implemented using an open source protocol known as OpenFlow[2-13].

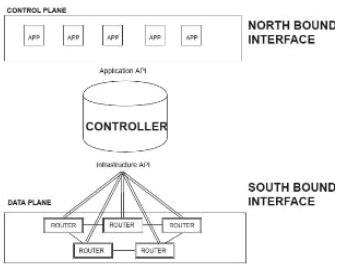

Figure 1: SDN Architecture

It is a network communication standard established between the controller and the data plane, and it configures the network devices and allows the controller to control network traffic, change routes dynamically and also determine the most optimal routes for packet flow. Each switch in a network contain a flow table that maintain the flows used to transfer a packet from a given source to a given destination. If a flow is not present in the flow table, the controller installs a new flow for the switch to utilize[13]. In addition, inactive flows are removed from the flow table periodically, as depicted in ***Figure 2***. As opposed to the traditional

network architecture where packet forwarding and updating of flow table were performed by individual switches, in SDN, updating flow table are handled by the controller, thereby delivering greater performance and faster results. It also provides flexibility and agility: two functions that lacked in the traditional network architecture.

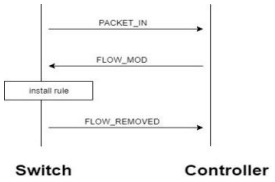

Figure 2: Flow Modification

Denial of Service

Due to the centralized structure of SDN, it can be an easy target for various malicious attacks such as Denial Of Service(DOS)[8]. DOS attacks[8] are one of the most common attacks when it comes to network security. The sole intent of the attacker is to disrupt the entire network or a particular machine, such as a server, by flooding it with spurious request packets, and preventing services from being rendered to users. DOS attacks[8]can easily be carried out as there are no prevention mechanisms so far[14]. However, measures can be taken to reduce the possibility of being a victim to a DOS attack[8]. In an SDN network, a DOS attack results in an unresponsive controller due to the constant spamming of spurious packets at a rate that exceeds its processing capacity. Since the controller is responsible for overseeing the transmission and the entire working of the network, the breakdown of the controller can completely paralyze the network.

This paper, centered around the concept of a DOS attack[8] on an SDN controller, provides an insight on the attack and how it can be ceased. As part of the solution, we have devised algorithms which can prevent the DOS attack from severely affecting the network.

2. Problem Statement

Ideal Case

DOS[8] is one of the most frequently occurring attacks that seize the network, temporarily suspending it. Even so, this attack does not have a definite prevention mechanism. DOS[8] attack basically causes the switch buffers to overflow due to the arrival of packets in excess[3]. This in turn leads to the degradation of the switch's performance and the gradual exhaustion of its buffering capacity[4]. Traditionally, when a packet arrives at a switch, the switch first refers its flow table to identify the flow to be used for transmission. If a flow exists, the packet is transmitted according to the flow and if a flow does not exist, the header part of the packet is extracted and forwarded to the controller for processing. But, when a switch buffer overflows, the packets are directly forwarded to the controller along

with its data part for further processing[3]. Processing packets containing data, is immensely time consuming and a cause of latency issues[9-10]. In other words, in an ideal SDN network, switches simply forward the received packet through the corresponding port to the corresponding adjacent switch based on the flow present in the flow table. If a flow does not exist in the flow table, the switch consults the controller by sending the header part of the packet, which is used by the controller to update the flow table, following which the packet transmission resumes.

DOS Attack

When a DOS attack[8] occurs in an SDN network, it first affects the concerned switch by overloading it with multiple request packets[4]. Each switch contain afixed queue that queue the arriving packets. When a switch is overloaded with multiple request packets, exceeding the queue's capacity, the switch suffers a breakdown and becomes incapable of processing the incoming packets. Hence, it begins to forward all the succeeding packets as a whole, i.e.: including both the header and the data part, to the controller[4].

The controller, however, contains a variable queue which queues the arriving packets and automatically expands or shrinks based on the amount of packets it holds. If all the packets begin to arrive at the controller in succession, the controller's queue will expand infinitely, degrading the controller's performance and efficiency, following which the controller may crash as its ability to process each packet and update the flow table decrease rapidly. Since the controller is the central point that controls the entire network[1], an unresponsive controller will result in single point failure thereby shutting down the entire network[11]. This problem will cause significant issues on services that are being provided to users, enterprises and other consumers. The objective here is to identify the DOS attack[8] and block the IP address of the attacker before it incapacitates the controller.

3. Solution

A DOS attack[8], due to its unpredictable nature, can easily strike any network and cause damage to its functioning. In the solution proposed, the controller queue is subjected to maintain a threshold value that monitors all incoming packets. ***Figure 3*** represents a basic SDN topology. As described previously, the controller contains a variable queue that queues all the arriving packets. This queue is assigned a threshold value, T1 to monitor the number of packets that arrive in the queue. A counter QC_i, where i=1,2,3..n, is set to check whether the packet count has crossed the threshold value. When the packet count of the controller queue, exceeds the threshold value, T1, the hashes of the packet's source and destination addresses are obtained from the hash table, appended to be stored in the CombHash table (refer ***Table 1***.) along with an individual counter, C_i, where i =1,2,3...n. For example, refer ***Figure 3***. If this hash combination is already present in the CombHash table, the counter, C_i, is simply incremented and if the combination is not present in the table, it is added as a new entry and then incremented. A threshold value, T2, is set for C_i and each C_i increment is followed

by a check to ensure that the count is within the threshold value. If this count if found to exceed T2, TTL(Time-to-live) of the current packet and all the packets identical to it in the queue, if present, is set to 0.

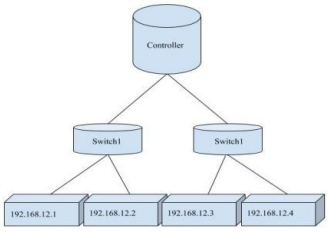

Figure 3: SDN Topology

Consequently, this minimizesthe controller's processing time when it encounters a packet with its TTL set to 0 as it immediately drops the packet and moves to the next one. Therefore, only those packets whose TTL is not 0 are processed by the controller. This process is performed successively for each packet in the controller queue from pkt_0 to pkt_T1. Once the TTL is set to 0 for all identical packets in the queue, the source IP of the packet is blocked to stop the spamming.The algorithms works on the basis of a timer. When the timer timeouts, the value of all C_i are cleared and reset to 0. Hence, it is assumed to be a DOS attack only if spurious packets arrive within the specified time frame and exceed the threshold value T2.

Table 1: Comb Hash Table

Hash combination	Counter
aea3c9ff44c4327e16bc3ef8136eb32aeb67925c9168cf02e190cc1bcb35cf69817d10219cf08197dc7783fdb62c6f589a91a88176e1726149cfc3ca82ca9e49	25
907107e9dc7b032d81aa58b2215ce239a6222f05044cde1e7c40a52c150e3754c12c9b234e0e6511da696dfb79e40a117eb0340260c4dcd766faf45d80ab3c2f	28
174e7a9e800db7f949ec4fb58c534be3b95cb64839af343d46344035c315c7985e6a1e3bf06c687c90c821d90c6c68052bb4c6fc7f4fff65ec7cb956760bc9a2	15
5369611bf5b6daf5863361e6127d70ca0082b5fafa1b1dd7d784224fa7e654d434377536ec572aa32b5dad992f3995dd88cd78a1c314062cf6a463140485c97c	6
89c49342f6665f44b77d8dec5a0da253750c6462d762ff8008fddf0ccf5ced53d9778a4c590270d0a443d6ee1e0c7493ed956ccc40eebbeab8593add8ffc0448	12

4. Algorithm

Set Threshold Algorithm

T1 : Threshold value for the controller queue
QC_i : Counter for packet count in queue ; i=1,2,3...n
Step 1: Start
Step 2: Set a minimum value for T1.
Step 3: If QC_i >=T1, then
3.1 : Repeat until queue is null
3.1.1 : Trigger GenHashScan Algorithm
End loop
End if
Step 4 : Stop

GenHashScan Algorithm

S_i : Source IP
D_i : Destination IP
CombHash Table : Contains #(S_iD_i), appended hashes of the source and destination IP of each packet.
C_i : Counter set for each hash combination ; i=1,2,3...n
Tm : Timer set to reset C_i to 0.
T2 : Threshold value of C_i.
TTL : Lifetime of a packet.
Step 1: Start
Step 2: Generate #S_i and #D_i
Step 3: Generate #(S_iD_i)
Step 4 : If #(S_iD_i) exists in the CombHash table, then
4.1 : Increment C_i
End if
Step 5 : Else
5.1 : Add #(S_iD_i) to the CombHash table
5.2 : Increment C_i
End else
Step 6 : If C_i>= T2
6.1 : For each packet with source id = S_i and destination id = D_i in the queue
6.1.1 : Set TTL=0
End if
Step 7: Stop

5. Conclusion

In this paper, we describe the functionalities of an SDN network and how a DOS attack can cause malicious effects on it. Further, have devised algorithms that can ease the constriction created in the network due to the attack and eventually stop the attacker from carrying out the attack. These algorithms work on the basis of threshold values and use a timer and hashing mechanism in detection phase. In the termination phase, the algorithm works with the TTL of the packets following which the attacker is blocked.

References

[1] Nunes B.A.A., Mendonca M., Nguyen X.N., Obraczka K., Turletti, T., A survey of software-defined networking: Past, present, and future of programmable networks, IEEE Communications Surveys & Tutorials 16(3) (2014), 1617-1634.

[2] McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J., Openflow: enabling innovation in campus networks, ACM SIGCOMM Computer Communication Review 38(2) (2008), 69–74.

[3] Dridi L., Zhani M.F., SDN-guard: Dos attacks mitigation in SDN networks, 5th IEEE International Conference on Cloud Networking (2016), 212-217.

[4] Kandoi R., Antikainen M., Denial-of-service attacks in OpenFlow SDN networks, IFIP/IEEE International Symposium on Integrated Network Management (2015), 1322-1326.

[5] Feamster N., Rexford J., Zegura E., The road to SDN: an intellectual history of programmable networks, ACM SIGCOMM Computer Communication Review 44(2) (2014), 87-98.

[6] Juniper Networks, Network Automation and Orchestration, Building an Agile Data Center Infrastructure with Juniper Networks (2015).

[7] Open Network Foundation, SDN Architecture (2014).

https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf

[8] Padmaja S., Vetriselvi V., Mitigation of switch-Dos in software defined network, International Conference on Information Communication and Embedded Systems (2016), 1-5.

[9] Kuźniar M., Perešíni P. Kostić D., What you need to know about SDN flow tables, International Conference on Passive and Active Network Measurement (2015), 347-359.

[10] Cisco, a platform to understand hardware and networking

https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-59/161-sdn.html

[11] Yu C., Lumezanu C., Sharma A., Xu Q., Jiang G., Madhyastha H.V., Software-defined latency monitoring in data center networks, International Conference on Passive and Active Network Measurement (2015), 360-372.

[12] Yan Q., Yu F.R., Gong Q., Li J., Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges, IEEE Communications Surveys & Tutorials 18(1) (2016), 602-622.

[13] Hu F., Hao Q., Bao K., A survey on software-defined network and openflow: From concept to implementation, IEEE Communications Surveys & Tutorials 16(4) (2014), 2181-2206.

[14] Nair A., BinyaMol M.G., Nair N.S., A mediator based dynamic server load balancing approach using SDN, International Journal of Control Theory and Applications 9(14) (2016), 6647-6652.