Detecting Denial-of-Service Attacks Using sFlow

Shivaraj Hublikar, Vijaya Eligar and Arun Kakhandki

Abstract This paper addresses how to detect denial-of-service attacks using sFlow. Denial-of-service (DoS) attack is a critical security challenge in software-defined network (SDN). In DoS attack, the network bandwidth is acquired by disrupting the services of the server by abruptly increasing the traffic and making the server unavailable for other users. The most challenging problem of DoS attack is to detect the attack almost instantly and in a precise manner. This paper presents the detection of DoS attacks by using sFlow analyzer, a SDNs flow monitoring tool. In the event of any attack, sFlow collects sample packets from network traffic, analyzes suspicious behavior and creates handling rules which are then sent to the controller. Implementation of DoS attack is carried out by emulating a typical network in Mininet and integrating this with sFlow analyzer. Through the simulated results, the potential DoS victims and attackers are quickly found.

Keywords Bandwidth detection · DoS attack · SDN · sFlow

1 Introduction

In a traditional data network, devices are structured into data-plane and control-plane which are local. If there are ten devices in a network, then each device has its own data-plane and control-plane that are having all the relevant information regarding forwarding tables. The data-plane comprises of switches, while the control-plane comprises of controllers of different types. Networking device will get the

S. Hublikar · V. Eligar (✉)
KLE Technological University, Hubballi, India
e-mail: vijayaeligar@bvb.edu

S. Hublikar
e-mail: shivaraj@bvb.edu

A. Kakhandki
KLSs VDRIT, Haliyal, India
e-mail: bvbarun@gmail.com

G. Ranganathan et al. (eds.), *Inventive Communication and Computational Technologies*, Lecture Notes in Networks and Systems 89,
https://doi.org/10.1007/978-981-15-0146-3_46

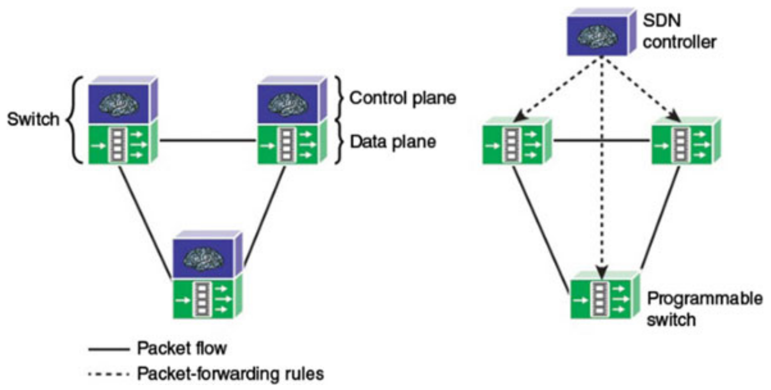


Fig. 1 Traditional network and SDN [11]

packets which are decided by forwarding tables. Since the demand of traditional networks is increasing, working on topology is complex [5]. Even though traditional networks are global and very popular, they have various drawbacks. Firstly, there is no scope to extend the network if a new feature or a protocol is to be added. Secondly, any new command cannot be accepted to improve the functionality of traditional networks since it is not programmable. Thirdly, cost of the network is effectively high since each device contains both data-plane and control-plane. Different topologies of the network are not possible since the traditional network is configured with set of predefined rules during the manufacture. Furthermore, the physical network infrastructure cannot be fully utilized since arranging the traditional network with predefined polices becomes complicated and error prone.

Recent trends such as machine learning, artificial intelligence, cyber security, internet of things (IoT) and mobile traffic have heavy traffic which cannot be managed by the network. SDN manages the overall network programmatically. SDN is very unique from the traditional networks which provide entire central control over the network by separating the control-plane and the data-plane as shown in Fig. 1.

SDN manages the network by centrally controlling the devices which greatly utilize and improve the network management. This network has data-plane within the device for sharing the forwarding data, whereas the control-plane is connected with separate device called controller which handles the information. SDN is categorized into three parts, controllers, southbound application program interfaces (APIs) and northbound (APIs). Controller, which is the programmable central system, controls the entire network which has the information of all the resources (like switches and routers) connected to this network. Switches and routers are the information devices which require southbound APIs as the medium for the controllers to transfer the data packets. OpenFlow is standard protocol used in southbound APIs. SDN uses northbound APIs to manage the traffic which is monitored by the network administrators to communicate with applications shown in Fig. 2.

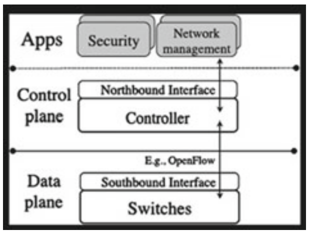


Fig. 2 SDN controller

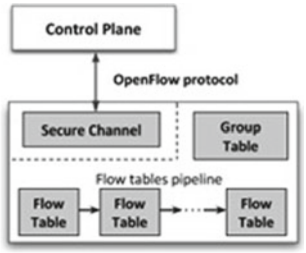


Fig. 3 Open switch architecture

1.1 OpenFlow

OpenFlow is a standard protocol that provides communication and interface between control-plane and data-plane in SDN. The data packets are transferred between devices which have to maintain traffic routing flows that is managed by the OpenFlow protocol. Each SDN device needs to maintain the set of Flow-tables that is controlled by OpenFlow switch. The incoming packets are controlled by the switch which are installed by the control-plane that maintains communication channel and also contains the Flow-table rules [1]. Figure 3 presents the logical structure of an OpenFlow switch. When there is a mismatch in the Flow-tables that does not match with any existing flow rules, the mismatch flows try to trigger forwarding plane to the controller which reduces the bandwidth and memory. The limited communication bandwidth between the control and data-planes could be a bottleneck of the whole network, and lead to security problems. Todays commercial OpenFlow switches only support cable connection to the controller. The practical connection bandwidth is tested to be less than 10 Mbps.

Research in SDN and DoS attacks is predominantly focused on detection of the attack. In July 2001, Internet infrastructure was hit worldwide by DoS attack worms named Code Red and NIMDA. Network scanning was used by Code Red Worm to attack the network at a rapid propagation rate to detect and exploit Internet Information Server (IIS) automatically. This DoS attack consumed huge bandwidth which affected lot of network devices. The attack was complex since the attack came from various sources of IP addresses for which packet analyzing was a big task. In order to mitigate or limit the damage to network resources, content filtering of some type were performed.

In [2], the authors propose a mechanism that avoids the overloading of switch TCAMs along with controller and control channel bandwidth and a solution called SLICOTS, which mitigates a type of flooding attack TCPSYN in SDN. It is built on top of the controller in order to monitor the network traffic related to TCP requests.

This paper is organized as follows. Section 2 discusses DoS attacks on SDN. Section 3 presents the DoS detection method based on sFlow tool. Section 4 demonstrates the method to detect DoS attacks. The simulation results and discussion are presented in Sect. 5 followed by conclusion in Sect. 6.

2 DoS Attack

DoS attack causes serious impact on the computing system. DoS attacks deny services to valid users by completely consuming the target resources. DoS attacks are usually initiated by an individual or group of individuals exploiting aspects of the Internet Protocol to deny other users from legitimate access to systems and information. The router hosts are disconnected if forwarding packets are stopped by router. The recent applications which are targets to these attacks are Web servers, mail servers and other services [6].

In [9], the authors explain SDN-aimed DoS attacks (data-to-control-plane saturation attacks). The attacker first sends the packets which do not match the packets in the Flow-tables by generating table-miss packets without the order with some or all fields, which does not match with existing flow rules on the target switch as shown in Fig. 4. Then, the attacker launches DoS attacks on the SDN network by flooding with large amount of table-miss packets. These table-miss packets will target massive packet in messages from the switch to the controller, and consume their communication bandwidth, CPU computation and memory in both control- and data-planes [2].

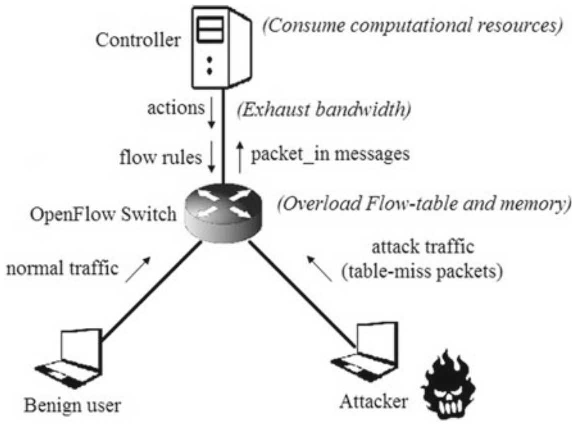


Fig. 4 DoS attack

DoS attacks can be of various types:

- **Destructive**: Attacks which destroy the device to prevent the proper operation of function, such as deleting or changing properties or information and power supply.
- **Resource consumption**: Attacks which try to reduce the ability of the device to perform effectively by opening many simultaneous connections to a single device.
- **Bandwidth consumption**: Attacks which attempt to affect the bandwidth capacity of the network device.

This paper aims to concentrate on bandwidth attack which is done using sFlow tool. When any DoS attack is detected, sFlow generates flow rules by analyzing samples of packets collected from the network traffic to be sent to the controller. In this work, security services are developed to protect networks against different type of network attacks for third parties like servers. A DoS attacker attacks the network by providing enormous flooding traffic in a short time to a server by increasing flow so that the server gets disconnected.

3 sFlow

sFlow helps to monitor the network, that develops various ways of handling the traffic flows, and to improve the performance of the network which consist of switches and routers. sFlow [12] is an open-source sampling tool used for measuring the traffic which is compatible with OpenFlow network. It consists of sFlow agents and collector. The sFlow agent captures traffic statistics from the device which is under observation that uses sampling technology. sFlow datagrams immediately forward the sampled traffic statistics to a sFlow collector for analysis. The main task of the two modules is listed below.

1. **sFlow Collector**: It is a server where sFlow datagrams are collected and stored.
2. **sFlow Analyzer**: It provides real-time overview of the network traffic flow by analyzing the received datagrams by analyzing the irregularities of the network parameters and detailed information. sFlow agents send a stream of sFlow samplings continuously to the collector where they are analyzed to supply a real-time, network-wide view of traffic flows.

3.1 Integration of Mininet and sFlow

Network simulators play an important role by evaluating network topologies of different types over a small scale. Many network simulators like Mininet, GNS3 and EsiNet are available. Mininet is used here which is an open-source network simulator used to generate traffic and analyze its flow.

Mininet [8] is an open-source network emulator and is a command line interface (CLI) and enabled simulator which supports the use of analysis tools like the sFlow, NetFlow and RMON. Mininet creates virtual switch, hosts, links and controllers on a single Linux kernel with a single command. Custom topologies are created using Mininet. It simulates a real machine and can create different hosts. The limitation of Mininet is that it does not have OpenFlow controller and it runs on slower links (10 or 100 Mbps).

According to [7], these simulators provide a platform to set a network topology as a replication to the real-world environment about analysis and detection of the attacks using Mininet and sFlow. The analysis is done in a number of steps in the tool, which are listed here.

1. Start.sh will run the shell script command to run the sFlow application.
2. In another window, Mininet topology will execute. The ping command is run to check the connectivity between the hosts of the topology created. A zero percentage drop depicts the complete connectivity between the hosts.
3. For accessing the sFlow trend GUI, a local host is created using the command: localhost: 8008.
4. A typical command in Mininet for detection of attack is given as: http://localhost:8008/app/mininet-dashboard/html/. It provides an approach to run a SDN Controller along with it.

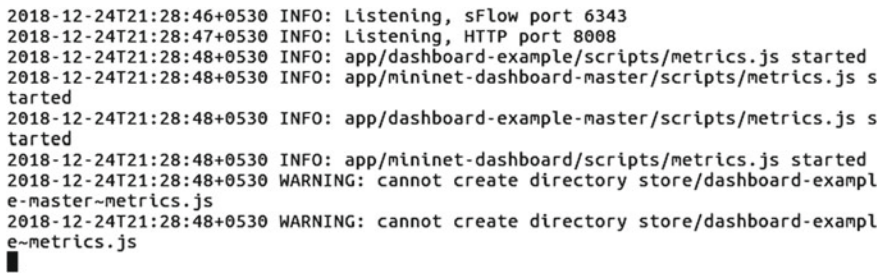
4 Implementation

In this section, the implementation of the network is discussed. Initially, sFlow-RT is created to receive a continuous flow of data that is sent from the network devices and converted into metrics. As soon as the flow reaches certain predefined metric level, it is sent to an analyzer. Next, using the Mininet command, a topology is built with link bandwidths of 10 Mbps. Finally, the output is the link between two hosts.

In order to make it easier to get started, the latest release of sFlow-RT includes a Mininet helper script sflow.py that automates sFlow configuration. The following example shows how to use the script and build a simple application in Python.

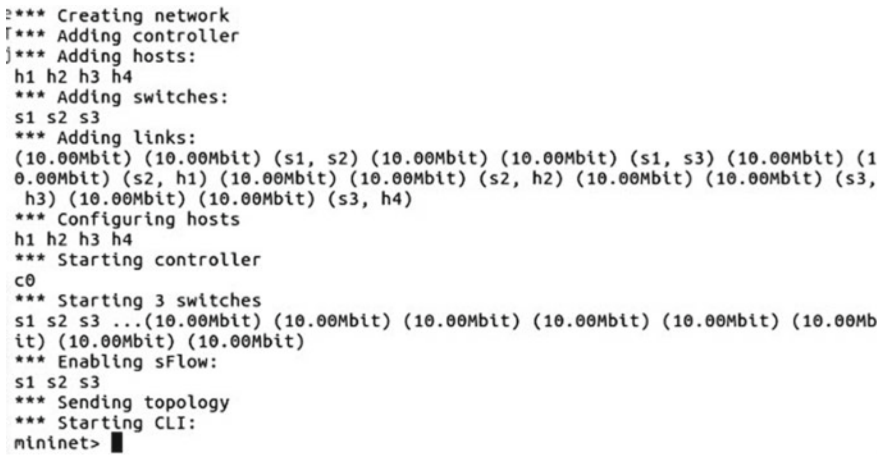
The various steps to detect the flows in a Mininet topology created in Linux environment using sFlow-RT are listed here and shown in Fig. 5.

1. **sFlow-RT**: sFlow-RT is an open-source tool that has an embedded OpenFlow controller, allowing monitoring and flow insertions to OpenFlow supporting switches. It analyzes certain events of interest, raise triggers and apply traffic handling rules to a particular controller. In order to analyze sFlow-RT flows and react on traffic changes, it has to be configured to work together with the existing network.
2. **For Creating Mininet**: In a second terminal, add the—custom argument to the Mininet command line. The following command builds a depth '2' tree topology with link bandwidths of 10 Mbit/s.
   cd sflow-rt
   sudo mn –custom extras/sflow.py –link tc, bw=10 –topo tree, depth=2, fanout=2.
   The response of the tool after creating the topology is shown in Fig. 6. The sflow.py



```
2018-12-24T21:28:46+0530 INFO: Listening, sFlow port 6343
2018-12-24T21:28:47+0530 INFO: Listening, HTTP port 8008
2018-12-24T21:28:48+0530 INFO: app/dashboard-example/scripts/metrics.js started
2018-12-24T21:28:48+0530 INFO: app/mininet-dashboard-master/scripts/metrics.js s
tarted
2018-12-24T21:28:48+0530 INFO: app/dashboard-example-master/scripts/metrics.js s
tarted
2018-12-24T21:28:48+0530 INFO: app/mininet-dashboard/scripts/metrics.js started
2018-12-24T21:28:48+0530 WARNING: cannot create directory store/dashboard-exampl
e-master~metrics.js
2018-12-24T21:28:48+0530 WARNING: cannot create directory store/dashboard-exampl
e~metrics.js
█
```

Fig. 5 Starting sFlow



```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(10.00Mbit) (10.00Mbit) (s1, s2) (10.00Mbit) (10.00Mbit) (s1, s3) (10.00Mbit) (1
0.00Mbit) (s2, h1) (10.00Mbit) (10.00Mbit) (s2, h2) (10.00Mbit) (10.00Mbit) (s3,
 h3) (10.00Mbit) (10.00Mbit) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...(10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mbit) (10.00Mb
it) (10.00Mbit) (10.00Mbit)
*** Enabling sFlow:
s1 s2 s3
*** Sending topology
*** Starting CLI:
mininet> █
```

Fig. 6 Creating topology

script extends Mininet, automatically enabling sFlow on each of the switches in the topology, and posting a JSON representation of the Mininet topology using sFlow-RT.

5 Results and Discussion

Whenever the network is simulated without any attack, bandwidth between the two hosts maintained at 9.63 Mbps. When a DOS attack is initiated on the network, the bandwidth falls to 30 kbps. The bandwidth has reduced from Mbits to kbits due to the DoS attack which has increased the datarate. This resulted in the decreased system performance. The GUI output of the DoS detection is shown in Fig. 7. Here, sFlow is initiated in the local host to detect the DoS attack when host h1 pings h2. As more number of packets are sent, DoS attack is detected.

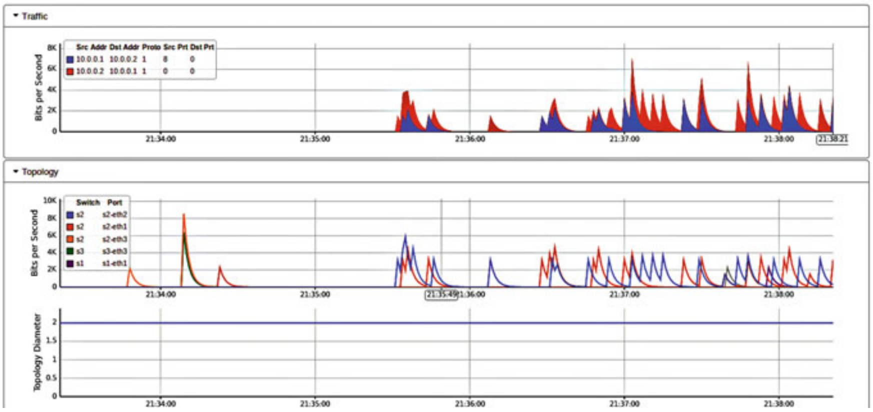


Fig. 7 DoS attack detected

6 Conclusion

DoS attack was detected by acquiring network bandwidth and disrupting the services of the server by abruptly increasing the traffic by making the server unavailable for other users. DoS attack was detected using the sFlow tool. In case of any attack, sFlow collects sample packets from network traffic, analyzes suspicious behavior and creates handling rules which are then sent to the controller. Implementation of DoS attack is carried out by emulating a typical network in Mininet and integrating this with sFlow analyzer. The results indicate efficient identification of DoS attack by using sFlow.

References

1. Ambrosin M, Conti M, De Gaspari F, Poovendran R (2017) Lineswitch: tackling control plane saturation attacks in software-defined networking. IEEE/ACM Trans Netw (TON) 25(2):1206–1219
2. Dridi L, Zhani MF (2018) A holistic approach to mitigating DOS attacks in SDN networks. Int J Netw Manag 28(1):e1996
3. Jyothirmai P, Raj JS, Smys S (2017) Secured self organizing network architecture in wireless personal networks. Wirel Pers Commun 96(4):5603–5620
4. Nugraha M, Paramita I, Musa A, Choi D, Cho B (2014) Utilizing OpenFlow and sFlow to detect and mitigate SYN flooding attack, 17(8):988–994
5. Ombase PM et al (2017) Survey on DOS attack challenges in software defined networking. Int J Comput App 975:8887
6. Othman RA (2000) Understanding the various types of denial of service attack. Bus Week Online. Accessed 12 Feb 2000
7. Peter: Mininet flow analytics. https://blog.sflow.com/2016/05/mininet-flow-analytics.html. Accessed 10 Jan 2019
8. Scarlato M. Network monitoring in software defined networking (thesis). Accessed 30 Jul 2014

9. Shang G, Zhe P, Bin X, Aiqun H, Kui R (2017) Flooddefender: protecting data and control plane resources under SDN-aimed DOS attacks. In: INFOCOM 2017-IEEE conference on computer communications. IEEE, pp 1–9
10. Sridhar S, Smys S (2016) A hybrid multilevel authentication scheme for private cloud environment. In: 2016 10th international conference on intelligent systems and control (ISCO). IEEE, pp 1–5
11. Stallings W (2015) Foundations of modern networking: SDN, NFV, QoE, IoT, and Cloud. Addison-Wesley Professional
12. Swapna AI, Reza MRH, Aion MK (2016) Security analysis of software defined wireless network monitoring with sFlow and FlowVisor. In: International conference on communication and electronics systems (ICCES). IEEE, pp 1–7