ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

A Framework for Design and Simulation of DoS Attacks on SDN Network

Charu P.P, Mary John

M.Tech Student, Dept. of I.T, Rajagiri School of Engineering and Technology, Cochin, India

Asst. Professor, Dept. of I.T, Rajagiri School of Engineering and Technology, Cochin, India

ABSTRACT: Software-Defined Networking (SDN) is an auspicious step towards the future network. SDN allows easily shape network traffic in real time. Researchers and number one IT companies have been embracing this evolution recently. The acceptance of a technology in an industry depends on its successful functioning in a real world scenario. The work described in this paper proposes a framework for design and simulation of the Denial-of-Service Attack against Software Defined Networks. In order to conduct the experimental research on SDN, an open source network emulator known as Mininet is used. A network of virtual hosts, controllers, switches, and links is created using the tool to simulate the scenario.

KEYWORDS: Software-defined network (SDN), Denial-of-Service Attack, Mininet.

I. INTRODUCTION

The emergence of computer networks opened a new portal of communication to the users. Networks facilitated users in various locations to be connected, and several technologies were further developed to enable the communication between the connected users. In 1999, the networks were essential for file transfers. The only thing under consideration during that time was regarding the size and speed of the network. Ie, how fast a file can be transferred from a point to another. Later in 2005, the development of Youtube, Skype, etc.. took the network to a whole new level. The VoIP (Voice over Internet Protocol) becomes a big deal after it allowed users to make and receive calls via the existing internet, in the same manner as it used to receive calls through the Public Switched Telephone Network.The Real-time communication considers an additional factor, latency, which is the time difference between a message sent and received. Thus, instead of considering transfer of a chunk of data between two points, the transfer of individual packets of data is considered.

Now, we deal with a networking environment where the system becomes more complicated. The explosion of mobile devices, advent of cloud services and server virtualization are among the trends driving the networking industry to reconsider traditional architectures of network. Now since more devices are connected to the network, different scenarios might prioritize a different parameter. Speed might be important in some cases, whereas some situation demands a better real-time communication. There is a need to dynamically configure information, and in these cases the SDN comes into play. Software Defined Networks [1] are used to separate physical equipment that store the data from their control mechanism. In this system, devices are present to store data and to manage the data flow. By doing this, the overall networking system much more manageable and intelligent. One control plane can be used to manage individual networking units such as routers and switches.As a result of new networking model which allows dynamic software-based control over packet routing, the routers and switches will no longer beexpensive, isolated, and proprietary hardware.

The Denial-of-Service (DoS) attack is the most common malicious attack. This attack grew more complicated with the advent of IP spoofing and mobile computing. Due to the ease of attack, anyone can bring down a website with a simple command prompt. Thecorporations and governments continue to explore better ways of attack prevention and detection.The agile acceptance of SDN creates new challenges regarding performance of security applications and scalability. In order to test new security applicationseffectively, a real-world Design and Simulation framework is necessary. The goal of this paper is to develop a Design and Simulationframework for DoS attacks on SDN networks. To achieve this goal we usean available open sourceSDN simulation tool, Mininet [4] and for DoS attack generation, IP flooding is used.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

II. RELATED WORK

Leveraging the contemporary research for large-scale networks, DoS attacks, and OpenFlow will provide valuable insight into an OpenFlow-security framework. Several recent papers and commercial applications are attempting to take advantage of the SDN/OpenFlow. Researchers from the University of Wurzburg have provided an early modeling and performance evaluation architecture for OpenFlow. Their work delivers a basic model for estimating packet sojourn time and the probability of lost packets, both of which are necessary to measure the effectiveness of OpenFlow against existing routing schema [2]. An OpenFlow study from Universidade Federal do Amazonas presented a method for DDoS attack detection based on traffic flow features, in which the extraction of such information is made with a very low overhead. Compared with traditional approaches, it provides some insight basic networking simulation and testing [3]. Understanding recent work from the University of Memphis on M&S utilizing game theory-based defense mechanisms against DDoS attacks provides a contemporary approach to the DDoS threat and M&S [4].

Commercial research also helps with DDoS attack simulation and detection techniques as well. For example, a project on BotNet detection by the Computer Science Laboratory at SRI International [5, 6], is one of a few early OpenFlow specific IDS-style approaches to DDoS. Additionally attack applications [7, 8], for current networking architecture will likely be helpful in the generation of attack message traffic for an OpenFlow network as well.

III. SOFTWARE-DEFINED NETWORK (SDN)

A. SDN/OpenFlow

SDN is amazing and we are now at the cutting edge of the SDN. The idea behind SDN is separating intelligence of the network away from the hardware. Back in the old days, if we had to store data on the network, a file server that had a hardware was present, where the data was configured. A single box is responsible for everything, the data, the hardware and the OS. If anything is to happen to the hardware, all the data would have been gone.Using SDN we can dynamically model network according to our wish. We separate networking into the control plane and into the data plane. Fig 1 shows the architecture of SDN. Data plane has all the routers and switches that allow packet to go around the network. Control plane contains set of management servers that communicate with the all networking equipment on the data plane and manage how should each data move from one point to another. To communicate between control plane and data plane protocol used is OpenFlow [9].

B. SDN – Working

SDN is an intelligent approach to building data networking software and equipment. The separation of the physical router or switch from control plane, creates an opportunity to a system administrator server to manage network flow decisions in near-real-time.

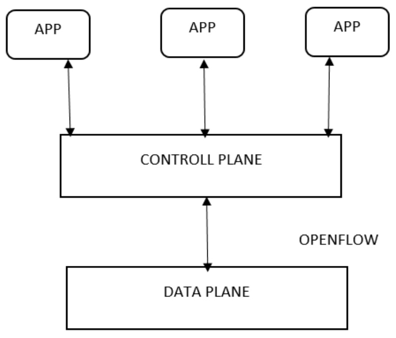


Fig.1. SDN architecture

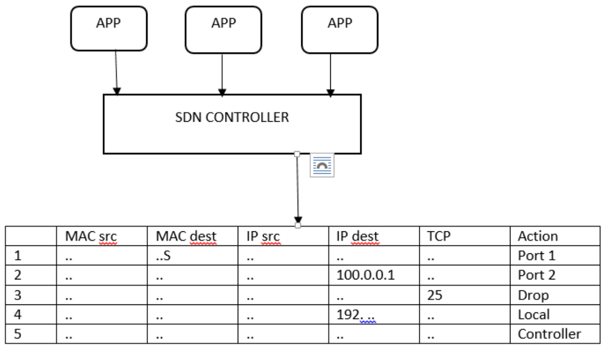
The data path of an OpenFlow switch consists of a Flow Table. There is an action associated with each flow entry and these set of actions is extensible. For our experiment purpose we consider a minimum requirement for all switches. The low-cost and high-performance the data path must have a carefully prescribed degree of flexibility.

Each flow-entry of OpenFlow switches has three basic action associated;

1. Forward this flow's packets to a given port. This allows packets to be routed through the network.
2. Encapsulate and forward this flow's packets to a controller. It is typically used for the first packet in a new flow, so a controller can decide if the flow should be added to the Flow Table.
3. Drop this flow's packets. This is used for security purpose, to control Denial-of-service attacks, or to reduce spurious broadcast discovery traffic from end-hosts.

A Flow-Table entry has three fields: (i) A packet header that is used to define the flow, (ii) The action, which defines how the packets should be processed, and (iii) Statistics, which keep track of the number of packets and bytes for each flow, and the time since the last packet matched the flow.

The Flow Table shown in Fig 2 explain four important functionalities of SDN. Entries one and two are standards switch routing based upon a given IP or MAC destination address. Entry three shows how OpenFlow can act as a firewall; drop all TCP from port 25. The entry number four shows the switch's ability to retain its own local logic. In this case, the switch knows traffic for IP address 192 should utilize local switch logic. The entry five represents a new flow in the switch, which was not previously encountered. For this entry switch should forward that packet to the SDN Controller for further processing.



	MAC src	MAC dest	IP src	IP dest	TCP	Action
1	..	..S	..	..	..	Port 1
2	..	..	..	100.0.0.1	..	Port 2
3	..	..	..	..	25	Drop
4	..	..	..	192. ..	..	Local
5	..	..	..	..	..	Controller

Fig.2. Flow table example

IV. DOS ATTACK

A DoS (Denial-of-Service) is an attempt to make a computer's resources unaccessible to its original user. If it is carefully planned and executed DoS attack can disable a computer and its networks. It can be mounted from anywhere to anywhere, at anytime, anyhow.In many shapes and forms DoS attack appears, and it can also have sub motifs. There are so many variables one can put on DoS attacks. A DoS attack can be perpetrated in a number of ways. The basic types of attack are:
1) Consumption of computational resources, such as disk space, bandwidth or processor time.
2) Disruption of configuration information, such as routing information.
3) Disruption of state information, such as unsolicited resetting of TCP sessions.
4) Disruption of physical network components.

A. Attack Generation

Ingesting any form of automated attack is a prime factor for the base line experiments. Here we used SYN flooding [12] to generate DoS attack. A SYN flooding is a simple denial-of-service attack. It takes advantage of the TCP three-way handshake model.

1) The attacker creates packets that contain spoofed IP-addresses; with every packet has a SYN flag set meaning it would like to open a new connection to the Server.

2) The Server receives the spoofed packets and is sending back ACK packets back to the spoofed IP-addresses. The Server will wait for an ACK coming back from the attacker, but since the IP-addresses where spoofed it will never get any packets back.
3) During this process the server connection table will be full, and all new connections will be ignored. This will affect all users who wants to make a connection to the Server.
4) After the attacker had his share of fun and stops flooding the Server, the Server normally goes back to its normal state.

V. DESIGN

A. *Implementation Technique*

When analysing network data traffics it found that if a user is frequent user he would sent at least 5 packet per connection to each destinations. If the user is abnormal or attacker transmit less than 5 packets per connection. We define n as the minimum number of packets per connection and m as the average number of connections of a frequent user.We assume that the characteristics of DoS addresses are to initiate less than m connections and to generate less than n packets per connection.

B. *The proposed method*

Here we define a table (T table) in the controller. The T table is used to store source IP addresses of forwarded packets. A counter c is used for each unique IP address which is used to track the number of forwarded packets from the switch.Whenever a new packet forwarded by the switch arrives at the controller, the controller assumes that it might be from the DoS attacking address firstly. The controller creates a new specific entry with hard timeout and idle timeout, of which values are smaller than those of normal entries, to limit its lifetime. Then, the source IP address is updated into T table for tracking, and its counter c is increased by 1 (Fig. 3). When c reaches m, we need to analyse its data traffic characteristic by requesting the average number of packet counters. If s is greater than n, this means the source address established and transmitted real data connections. In other words, it is a frequent user. Hence, the controller issues a modification message to reset all hard timeout and idle timeout of its existing entries to normal value. In vice versa, if s is smaller than n, this is malicious traffic. The controller dispatches a dropped rule for the address to the switch.

C. Simulation Tool Used

Mininet is a virtual network environment that can run on a single PC. Mininet is particularly usefull for experimentation and learning. Mininet run on real kernel, switch and application code on a single machine. It provides command line, Grapical user Interface and python Interface. Many openflow features are already built in to the environment. We can create virtual network in Mininet and we can connect in real openflow controller to real software open flow switches they are running in the virtual network environment. Why use Mininet
Fast: it is very easy to set up network using mininet.
Using Mininet it is possible to create custom topology
It can run real programs (anything that can run on Linux can run on a Mininet host).
It support Programmable OpenFlow switches.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

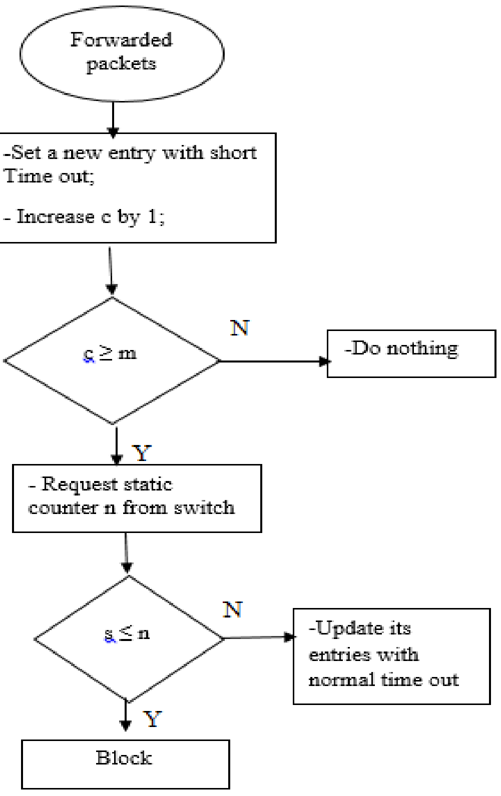

Forwarded packets
-Set a new entry with short Time out;
- Increase c by 1;
c ≥ m
N
-Do nothing
Y
- Request static counter n from switch
s ≤ n
N
-Update its entries with normal time out
Y
Block

Fig.3. Workflow

It is possible to use mininet in two different design alternatives. It can deploy on a real system such as real servers, real switches.Deploying Mininet in real system make it possible to reconfigure their topologies because it is on software.In other way Mininet can design using network virtual machines. This makes it possible to connect virtual machines running on machine and network them together with software switch that running on the machine.

Fig 4 illustrate how Mininet works in a nut shell. When run Mininet it will run nm which is a launcher for Mininet process .that control some process each of which represents single host running on in the network .for example want to create a network with two hosts the Mininet launcher would launch two dash processes and then it would create a unique namespace for each of the processes. Next thing happens is Mininet creates virtual Ethernet pair, this is used to create tunnel between the virtual interface and real interface. Once those Ethernet pairs are created and assigned to namespaces then create OpenFlow switch to connect those hosts. Then finally can instantiate an OpenFlow controller,that use Openflow protocol to control the flow entries in the switch. The entries in the switch allows packets to travel between each of these hosts in the virtual network.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 2, February 2016

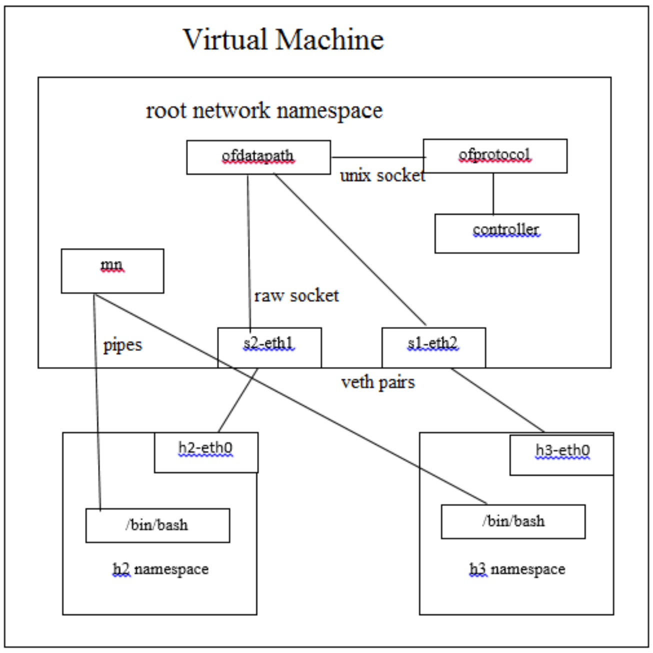


Fig.4. Mininet VM in a Nutshell

To simulate the proposed method, a small topology is simulated using Mininet. Which consists of 1 webserver and 3 PC clients (1 DoS attacking user, 1 malicious user and 1 frequent user) connected to the SDN network (Fig. 5).

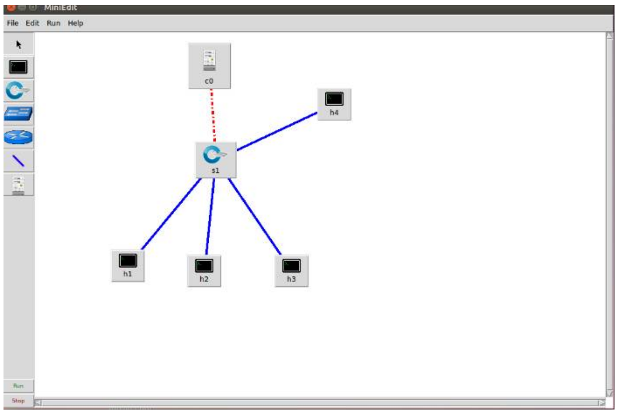


Fig.5. Network topology created using Mininet

• In the switch, the flow table has capacity of 10.000entries.Initially, it is empty. The normal entry has hard timeout and idle timeout equal to 600 seconds and 60 seconds. The entry for DoS attacking user has hard timeout and idle timeout equal to (60, 10) seconds.

• The malicious user has IP address 10.0.0.1. It injects spoofed packets to the switch infinitely. For each packet, the destination IP address is generated randomly.

• The DoS attacking user sends spoofed packets to the switch infinitely. For each packet, the source and destination IP addresses are generated randomly.

• The frequent user has IP address 10.0.0.2. It establishes 5 different connections to the server, and transmits 10 packets per connections.

VI. CONCLUSION AND FUTURE WORK

Software Defined Network replaces the traditional network by allowing a software to control and manage the various devices within the network. Allowing a software to control a network adds various benefits in terms of accessibility by providing an abstract to the data plane and control planes. A Denial-of-Service attack is one which does not allow a user to access resources that are originally accessible to the user. When denied service, the overhead of SDN Controller increases. In this paper, a feasible method to combat the DDoS flooding attack is proposed. The simulation tool used in the work is Mininet, that has a GUI and Python interface to make it user friendly. An issue arises with the DDoS attack when the traffic is huge. The drawback needs to be addressed in my future work, where the technique needs to be optimized to be able to adapt to varying environment with various types of protocols.

REFERENCES

[1] S. Agarwal, M. Kodialam, and T. V. Lakshman, “Traffic engineering in software defined networks,” in Proc. IEEE INFOCOM, Apr. 2013,pp. 2211–2219.
[2] M. Jarschel, et al. “Modeling and performance evaluation of an OpenFlow architecture.” In *Teletraffic Congress (ITC), 2011 23rd International,* Sept. 2011.
[3] R. Braga, et al. "Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow," In *Proceedings of 35th Annual IEEE Conference on Local Computer Networks*, Denver, CO, 2010.
[4] Q. Wu, et al. “On Modeling and Simulation of Game Theory-based Defense Mechanisms against DoS and DDoS Attacks.” In *Proceedings of the 2010 Spring Simulation Multi-conference.* 2010.
[5] “BotHunter: A Network-based Botnet Diagnosis System.” (Apri. 2013). [Online]. Available: http://www.bothunter.net/ .
[6] "OpenFlow Security." (May 2012). [Online]. Available: http://www.openflowsec.org/OpenFlow_Security/Home.html.
[7] “StormSecurity: IT Security Research and Services” (Mar. 3, 2009). [Online]. Available: http://stormsecurity.wordpress.com/2009/03/03/application-layer-ddos-simulator/.
[8] Tools Yard: Hacking Tools and Penetration Testing Tools.” (Jun. 2012). [Online]. Available: http://toolsyard.thehackernews.com/2012/06/bonesi-ddos-botnet-simulator.html.
[9] N. McKeown, et al. "OpenFlow: Enabling Innovation in Campus Networks," In SIGCOMM Computer Communications Review, vol.38,no. 2, 2008.
[10] A. Way. (2013). "Alan Talks Tech." [Online]. Available: http://alantestwiki.pbworks.com/.
[11] “Mininet – An Instant Virtual Network on your Laptop.” (2014). [Online]. Available: http://mininet.org/.
[12] SYN flood. Available:https://en.wikipedia.org/wiki/SYN_flood
[13] “OpenFlow Switch Specification Version 1.3.4,” ONF Specifications, Open Networking Foundation, 2014.
[14] Shin, Myung-Ki, Ki-Hyuk Nam, and Hyoung-Jun Kim."Software-defined networking (SDN): A reference architecture and open APIs." ICT Convergence (ICTC), 2012 International Conference on. IEEE, 2012.
[15] Alcorn, Joshua, and C. Edward Chow. "A framework for large-scale modeling and simulation of attacks on an OpenFlow network." Computer Communication and Networks (ICCCN), 2014 23rd International Conference on. IEEE, 2014.
[16] Wang, Haopei, Lei Xu, and Guofei Gu. "FloodGuard: a dos attack prevention extension in software-defined networks." Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on. IEEE, 2015.
[17] Dao, Nhu-Ngoc, et al. "A feasible method to combat against DDoS attack in SDN network." Information Networking (ICOIN), 2015 International Conference on. IEEE, 2015.
[18] Lantz, Bob, Brandon Heller, and Nick McKeown. "A network in a laptop: rapid prototyping for software-defined networks." Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010.
[19] T. Peng, C. Leckie, and K. Ramamohanarao, “Protection from distributed denial of service attacks using history-based IP filtering,” in Proc. IEEE ICC, 2003.
[20] S. Scott-Hayward, G. O’Callaghan, and S. Sezer, “SDN security:a survey,” in Proc. IEEE SDN4FNS, 2013.
[21] OpenFlow Firewall Application. https://github.com/hip2b2/poxstuff/blob/master/of firewall.py.
[22] OpenFlow Specification v1.4.0. http://www.opennetworking.org/images/stories/downloads/sdn-resources/onf specifications/openflow/openflow-spec-v1.4.0.pdf.
[23] OpenWrt: a Linux https://openwrt.org/.