International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 14 (2015) pp 34294-34301

Protection of sdn by implementing firewall policies in controller

Manasa.S, Prof. Manivannan.S.S*

M.Tech (IT), School of Information Technology, VIT University, Vellore-632014, Tamilnadu, India

**Assistant Professor (Senior), School of Information Technology, VIT University, Vellore-632014, Tamilnadu, India*

*E-mail: manasa.jamuna@gmail.com, *manivannan.ss@vit.ac.in*

Abstract- Software-Defined Networking (SDN) is an emerging technology in the networking domain. This technology decouples the control plane and forwarding plane from the network elements such as routers and switches. The decoupling of two planes helps to develop more advanced and complex applications. Convention firewall fails to provide a strong protection against the attacks specifically against Distributed Denial-of-Service (DDoS) attacks on SDN. Existing firewall is dedicated hardware, more expensive, single point of failure, static and Access Control List (ACL) based firewall. All these features of existing firewall make it unsuitable for securing SDN. In our paper, we proposed distributed and dynamic Firewall system called SDN Firewall. SDN Firewall overcomes all the disadvantages of conventional firewall. SDN Firewall is implemented at POX Controller using 3 popular firewall techniques – Packet Filter Firewall, Stateful Inspection Firewall and Application Proxy. Analysis of results obtained from emulation of SDN Firewall in Mininet emulator shows that SDN Firewall is more secure than conventional firewall system.

Keywords- SDN, DDoS, Packet Filter, Stateful Inspection, Application Proxy, Mininet and POX Controller.

1. Introduction

SDN is one of the utmost important technologies network industry has noticed in recent times. The emerging SDN satisfies all the new network requirements in the areas such as big data, cloud computing and mobility applications. SDN architecture is cost effective, dynamic and adaptable and is considered as ideal for high bandwidth applications. This architecture separates forwarding and control functions. The control plane can be programmed directly and infrastructure is abstracted for network services and applications. SDN architecture has data, controller, and application planes. Data plane consists of network elements and performs processing function and traffic forwarding function. Application plane consists of SDN applications. Controller plane exists at the center of the SDN architecture and consists of controllers and its function is to translate requirements of the applications and executes a low level control on the network elements. SDN architecture is directly programmable, agile, centrally managed, programmatically configured and open standards-based and vendor-neutral [7].

The SDN Firewall is developed in Network Emulator called Mininet [19]. Mininet is open source product. Mininet provides strong Python API for virtual network creation. Isolated hosts, Emulated links and Emulated switches are the main mininet network components. Mininet's CLI and API are used for interaction with virtual network. Mininet uses Process-based Virtualization to run switches and hosts on single real kernel and all these processes run in Mininet's network namespace.

The entire SDN network is centrally controlled by POX Controller. POX is python-based Open Flow controller and is sibling of NOX which is C++ based. Officially POX requires Python 2.7 for proper working. POX is open source is hosted by Git Hub repository [13]. Open Flow is standard which controls the packet forwarding in SDN switches. It has an open protocol called Open Flow Protocol and is used for programming switch's flow table. Switch's Flow table, secure channel and Open Flow Protocol is organized by Open Flow. Flow entries in the Flow table of a switch decide whether the incoming packets at the switch should be dropped or forwarded. Secure channel established between POX Controller and Open Flow switches is used for communication between them. Open Flow Protocol possesses Open Flow messages which are utilized for communicating between Open Flow switches and POX Controller [8].

2. Literature survey

Bilal, Muhammad, Maqsood and Khaled performed 2 types of structural analysis such as Relational Algebra and Policy Tree, on the firewall rule set structure to identify and to resolve the existing conflicts in the rule set [1].

Chandan, Veena, Ram and Murthy developed a custom topology framework which helps in creation of user required custom network topology. The framework will not affect existing Mininet performance [2].

Sethi, Srinivas and Sharad have described formal verification of SDN controller using network state and data state abstractions for Model checking for large count of packets exchanged between the hosts in the network [3].

David Meyer discussed about three design points in SDN currently under research SDNRG. The three design points are distributed versus centralized control, various levels of separation of data and control planes and choosing to make either control plane programmable or data plane programmable [4].

Errin proposed 2 types of parallel firewall architectures in high speed networks. Data parallel firewall architecture distributes the packets arrived across the firewall array to achieve higher throughput and function parallel firewall architecture distributes the firewall rules across the firewall array to reduce processing delay [5].

Gopal, Amaresh, Mandal and Bhargab proposed an efficient method of packet filtering firewall based on Binary Decision Diagram (BDD) which consumes less memory and less time duration for rejecting or accepting the packets [6].

Hiroaki presented a detailed study on requirements of a switch used in SDN. Programming flexibility

requirement at switch should allow programmers to write their own logic in the action field of flow table and Packet Processor executes programs in action field [7].

Kim, Jaebeom and Young suggested an alternative testbed for realization of SDN called Raspberry-Pi. This testbed is Linux machine which reduces SDN implementation cost effectively compared to other testbeds such as NetFPGA and Mininet. But throughput of this Linux machine is comparable with NetFPGA-1G [8].

Ian and Martin proposed firewall security system that effectively reduces the cost of packet matching as compared to traditional firewall by ordering the rule set, that is, by placing the most frequently matched rule at the top of the rule set [9].

Iman, Maryam and Ali made a detailed survey on classifying the vulnerabilities of firewalls based on the nature of vulnerabilities. This survey creates a better perspective for future research in this field. They also discussed about firewall fingerprinting which helps attackers to gather more precise information on vulnerabilities in firewalls [10].

Jake and Jun proposed hardware firewall implemented in SDN environment and are stateful. The major limitation faced in this hardware firewall is communication overhead between a controller and a switch which results in high latency for the flows [11].

Kavin and Akharin proposed a system which uses Firewall rules in routing. The main aim of this paper is to learn the firewall policies used at the destination using BGP routing protocol and to reduce bandwidth consumption [12].

Ligia, Christiane, Ailton and Rogerio demonstrated difficulties faced in computer networks domain currently to satisfy the requirements of the Internet applications and how SDN is considered as one of the promising approaches for Future Internet [13].

Liu, Wen and Wang proposed a SDNoC in which network is programmable and configurable on the chip using SDN technology. In SDNoC, control logic is separated from the chip hardware and the applications configure the network based on their own requirements. The results of simulation shows SDNoC successfully reduce energy consumption and improve the network performance [14].

Mounira, Pujolle, Ahmed, fadlallah and Fouad described 3 approaches for realization of network virtualization such as OpenFlow, 4WARD and AKARI which give a platform for validation of on-demand virtual network architecture. Among the 3 approaches, Open Flow provides satisfactory simulation results for On-demand virtual network in terms of programmability, scalability and virtualization [15].

Michelle, Hyong, Lee and Yang developed firewall application implemented on POX controller that possesses all the features of firewall hardware. The developed application performs a comparison between packet headers and firewall rules in order of priority from highest to lowest. This approach mainly reduces the firewall hardware cost and maintenance cost [16].

Luo and Chen proposed a solution to the challenges faced by cloud systems which comprises several datacenters. Network abstraction to physical and virtual data plane in a datacenter and network primitive to solve interconnection issues across several datacenters are the 2 effective techniques proposed. Performance results proved SDN as a promising and effective solution to cloud systems [17].

Rihab and Lamia proposed a shortest path routing solution in SDN environment using POX as a SDN Controller. Topo_proactive POX Component is used to identify the shortest path in the network [18].

Rogerio, Ailton, Christiane and Ligia provided a brief description of SDN elements such as programmable switches, controllers and OpenFlow protocol between the above 2 elements. The main aim is to evaluate the Mininet emulation tool for SDN architecture. PING and scalability tests concluded that Mininet tool works efficiently at faster speed and it is valuable tool for research in SDN [19].

Ruxandra, Mihai, Razvan and Nicolae presented a paper on ACL which is centralized. They developed an application using Cisco's onePK framework called FirewallPk. The application performs inspection on the real-time traffic and installs appropriate rules to allow or reject the packets in a centralized fashion which frees firewall configuration from human errors [20].

Suchart, Atipong and Somnuk proposed new firewall design technique which consists of 2 major parts, Single Domain Decision firewall (SDD) and Binary Tree Firewall (BTF). The former part eliminates the conflicts in firewall rule set and the latter part is a data structure that increases the firewall rules checking process [21].

Safae, Amina, Driss, Rachid, Mohamed and Jaafar proposed a new SDN architecture which is decentralized and dynamic in nature. The proposed SDN architecture possesses self-organizing capability to adapt to changing environment and it is decentralized i.e. hierarchy of controllers which overcomes reliability and scalability issues of centralized SDN [22].

Wang compared and evaluated the scalability, correctness and performance of OpenFlow Network Emulators such as EstiNet and Mininet and OpenFlow Network Simulator, EstiNet in grid network environment. When the ping program is run, Mininet consumed more time for launching program, setup and while releasing resources as network size increased. But EstiNet emulator provided good performance results, consumed less time and scalable. EstiNet simulator require more simulation time [23].

Latifi, Arjan and Cico developed a network topology in Mininet and ns-3 with same number of network devices. The network topology built in Mininet is based on SDN technology and in ns-3 is conventional network. The tests results of the parameters such as delay and convergence time obtained from Mininet network model outperforms the conventional model [24].

Theuns and Ray Hunt demonstrated how the conventional firewall system results in autoblocking which further leads to Denial of Service. This drawback is overcome by packet filtering system. Prototype consists of iptables along with 2 modules, kernel bag and condition match/target. Prototype can support complex security policies that are difficult in case of convention firewall techniques [25].

3. Proposed system

A. Proposed System Architecture

The proposed SDN architecture consists of 6 OpenFlow switches, 8 hosts and a POX Controller as shown Fig. 1. All the switches in the system are directly controlled by central

POX Controller. The hosts are not under the direct control of POX Controller and are not connected to the POX instead they are directly connected to leaf switches. Entire traffic in the network is managed by the POX Controller. Communication between OpenFlow switches and controller happens over a secure control channel established between them. The communication protocol is popularly known as "OpenFlow protocol".

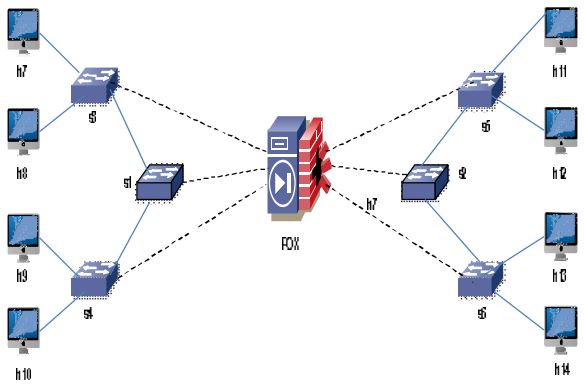

Fig. 1 Proposed SDN Architecture

Every Open Flow switch in the network consist a single flow table. The flow table consists of flow entries. Each entry has header fields to match against the packet, counters which keep track activities and actions to apply on the packets. Actions are applied to only those packets which match the flow entries in the flow table. When a switch receives packets, it compares the packets fields against flow table. If there is any matching entry found then corresponding actions are executed. The action can be forwarding the packet to a port specified or it can be dropping the packet. If there is no match then packet is forwarded to POX Controller using Openflow protocol over the established secure channel. Packet is now handled by POX controller which parses the packet and installs the flow rule in the switch's flow table.

In SDN architecture, all the Open Flow switches form Data Plane, POX controller corresponds to Controller Plane and Firewall Application in the above proposed system architecture exists in Application Plane. The proposed system implements distributed and dynamic Firewall application. Distributed Firewall means enforcing or execution of firewall policies is distributed to all the Open Flow switches. But managing the firewall policies is centralized at the POX Controller. All the addition, deletion and updating of firewall policies is responsibility of central POX controller. Dynamic firewall means firewall rules are added to flow table dynamically by monitoring the network traffic. POX Controller monitors the traffic, whenever the attack is identified firewall rules are added to flow table of the particular Open Flow switch where the attack is detected. The distributed and dynamic firewall system overcomes the challenges faced by the conventional firewalls.

B. Proposed SDN Firewall Design

In Fig. 2, a secure control channel is established between Open Flow Switch and POX Controller then switch is said to be connected to controller. OpenFlow messages are exchanged between switch and POX Controller over the new connection. In the proposed design, distributed and dynamic firewall application is started at the controller which monitors the network traffic for attacks. When the attack is found, firewall rule to block the attack is generated at controller and is send to

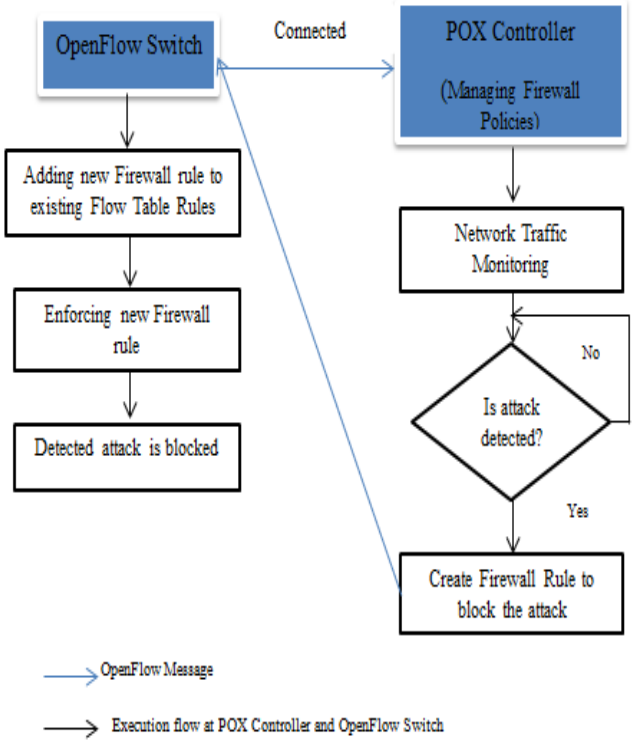

Fig. 2 Workflow of SDN Firewall

Open Flow switch where the attack is detected. Open Flow switch receives the message from controller and adds the new firewall rule to exiting flow table at the switch. Newly added firewall rule blocks completely the attack found providing strong security mechanism to Software Defined Network. In this design, all firewall rules are managed centrally at controller but the actual enforcement of firewall rule takes place at the OpenFlow switch. Hence the firewall application is said to be distributed at the OpenFlow switches. All firewall rules are created dynamically mainly based the network traffic.

C. SDN Firewall Techniques

1. Packet Filter Firewall

The Packet Filter Firewall monitors the state of network traffic and uses this information to determine which network packets to allow through the firewall. By recording session information such as IP address and protocol, a dynamic packet filter implements a much stronger security than a static packet filter. The implemented dynamic packet filter is distributed in nature i.e. packet filtering rules are all generated and distributed to all switches in the network to enforce the filtering policies by the centralized POX Controller [26].

i. Ping Flood Attack and Mitigation Technique

A ping flood is a basic and simple denial-of-service attack. In this attack, the attacker overwhelms the servers with ICMP Echo Request packets. This attack is uses ping command's flood option. Flood option when used with ping, sends a large number of ICMP packets without waiting for ICMP Echo Reply packets at a faster speed. The client hopes that the victim will respond with ICMP Echo Reply packets and results in bandwidth consumption of both incoming and outgoing. If the victim system runs slowly, there are high chances that ping flood option consumes CPU cycles also. These changes are noticed by a significant slowdown of the target victim which is experienced by the user.

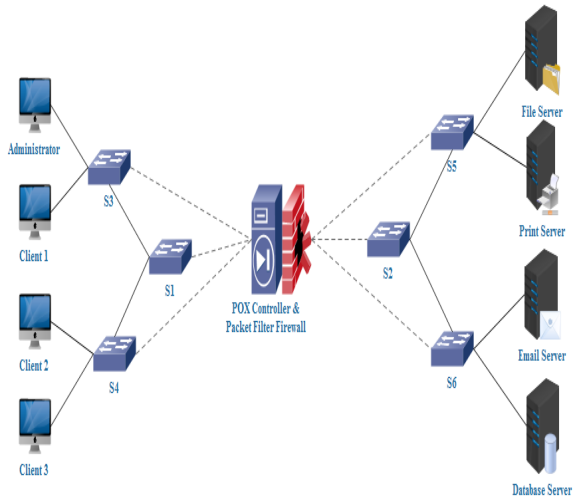

Fig. 3 Packet Filter Firewall in POX

Packet filtering firewall module implemented filters out the ICMP Traffic generated using by "PING" command. This firewall detects and blocks the "PING FLOOD ATTACK" from the clients to servers in the network. In the Fig. 3, firewall blocks the ping flood attack from Administrator and 3 clients to the 4 servers. This also blocks the ping flood attack on Administrator by the 3 clients.

ii. Commands

Mininet - <Source host> ping <options> <Destination host>
For Ping Flood Attack, -f option is used - <Source host> ping -f <Destination host>

2. Stateful Inspection Firewall

Stateful inspection firewall technique performs monitoring function on the state of all active connections and stores the information in a state table. This obtained information is used for making decisions whether packets should be allowed through the firewall or rejected. This technique is popularly known as dynamic packet filtering. Apart from the filtering function, this technique also tracks the active connections on all interfaces of firewall. The tracked information is used for validity checking of every active connection [26]

i. Connection Flood Attack and Mitigation Technique

Connection flood is an effective attack. In this attack, a client floods the state table by starting or initiating large number of new TCP connections but do not release the established TCP connections for a long time. This causes new server connections from other clients into WAIT state. This state results in efficiency decrease and even exhausting server resources. The final result is an inability to respond to the connections initiated by other clients.

Stateful inspection firewall implemented overcomes connection's state table from overflowing by "CONNECTION FLOOD ATTACK". This firewall installs the firewall rule in the flow table of switch to allow maximum 2 new connections from the same client at once. Further new connection requests from the same client are blocked for time duration of 25 seconds called hard-timeout. After this hard-timeout, same client is allowed to establish limited number of new connection to the web server and download the required files. This firewall technology gives equal opportunities for all the clients in the network to utilize server resources efficiently.

In the Fig. 4, stateful inspection firewall is implemented at the POX Controller. This firewall successfully mitigates the connection flood attack and provides equal opportunity to access the web server to administrator and 3 clients.

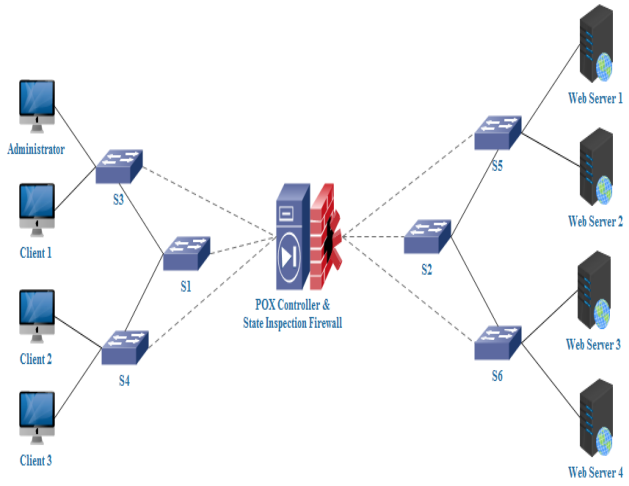

Fig. 4 State Inspection Firewall in POX

ii. Commands

- To start HTTP Web Server: $ python –m Simple HTTP Server 80
- To download files from the HTTP Web Server to Client Machine: $ wget http://<ip address of HTTP Web Server>:80/<filename>

3. Application Proxy Firewall

i. Custom Topology Creation and Internet Connectivity

- Adding External Remote controller to the network.
- Create 3 OpenFlow vSwitches and 4 hosts i.e. 1 Administrator and 3 clients and establish links between them as shown in the Fig. 5.
- Connecting network to root namespace.
  - Create a node in the Root Namespace called as Root.
  - Prevent Network Manager from interfering with our network interfaces.

- Create link between every switch in the network to the Root.
- Start network that now includes link to root namespace.

- Configure NAT using iptables which consist of chains in the Linux Kernel. Each chain is a list of packet filter rules.
- Start NAT and instruct the kernel to perform forwarding using sysctl which is used to modify kernel parameters at runtime.
- Establish routes from every end hosts in the network and set root as default gateway.
- In order to stop NAT,
  - Flush any currently active rules.
  - Instruct the kernel to stop forwarding.

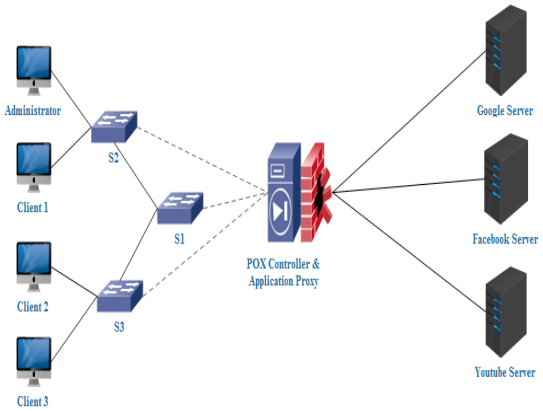

Fig. 5 Application Proxy Firewall in POX

ii. Blocking Access to Particular Sites

- Administrator or 3 clients are prompted for URL to which he/she needs to access.
- If Administrator or 3 clients enters following 2 URL's i.e. http://www.facebook.com and http://www.youtube.com", they receive "Access Blocked to this Site" response

4. Results and discussion

A. Packet Filter Firewall

In the below screenshots, h7 is Administrator, h8, h9 and h10 are clients. h11 is File Server, h12 is Print Server, h13 is Email Server and h14 is Database Server.

- Ping Flood attack by h7 on 4 servers is blocked by the Packet Filter firewall.

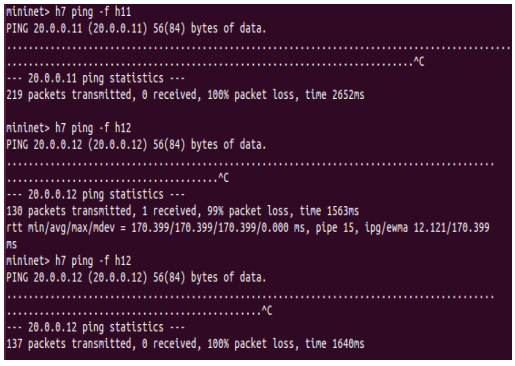

Fig. 6 Blocking Ping flood attack by h7 on h11 and h12

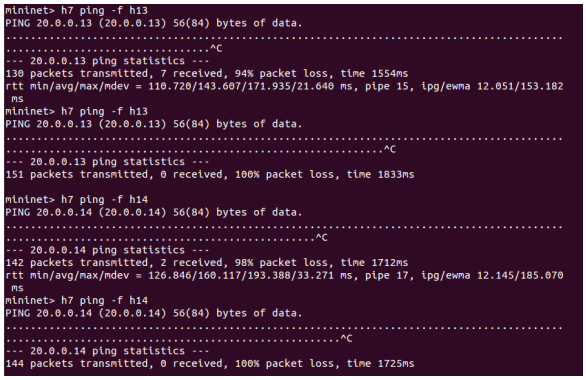

Fig. 7 Blocking Ping flood attack by h7 on h13 and h14

B. Stateful Inspection Firewall

In the below screenshots, h7 is Administrator, h8, h9 and h10 are clients. h11, h12, h13 and h14 are HTTP Web Servers.

- After downloading hello.txt continuously from h11 web server, h7 web request is serviced by h11 after 25 seconds of hard timeout.
-

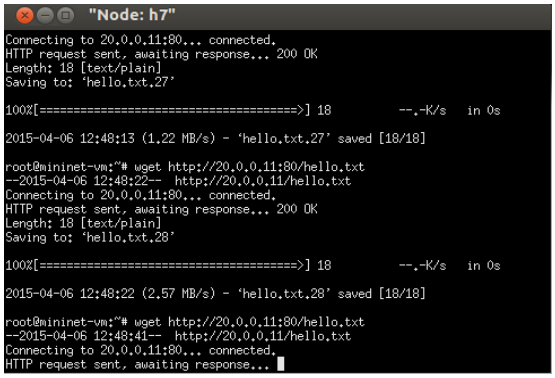

Fig. 8 File Download by h7 from web server h11

C. Application Proxy Firewall

In the below screenshots, h4 is Administrator, h5, h6 and h7 are clients. There are 3 servers namely, Google server, Facebook server and Youtube server.

- Access to Facebook and Youtube server is successfully blocked for all hosts. Below screenshot shows access blocked for host h7.

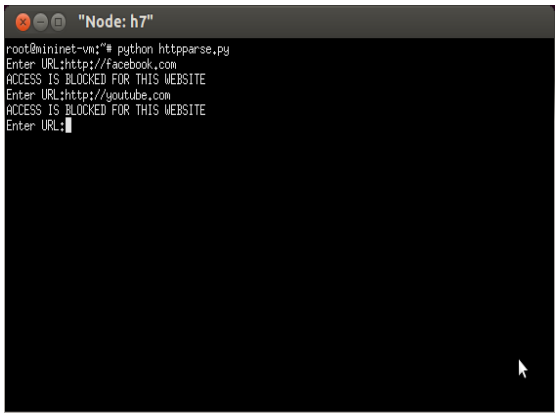

Fig. 9 Access blocked to Facebook and Youtube servers

- Access to Google server is allowed for all hosts (Administrator and 3 clients). Below screenshot shows Google server access by host h4.
-

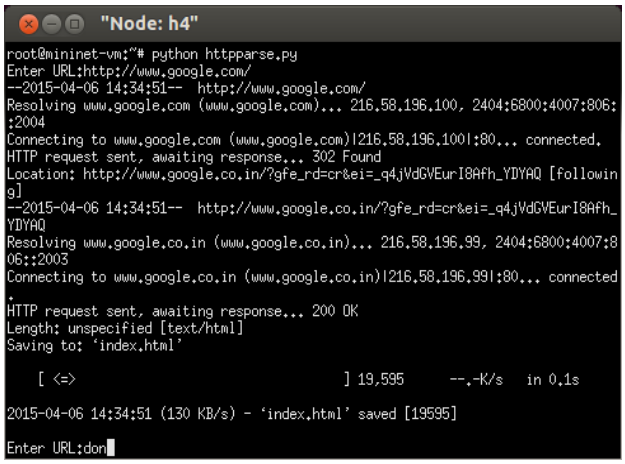

Fig. 10 Access to Google server

D. Comparison between SDN Firewall and Existing Firewall

Table 1 Comparison between SDN Firewall and Existing Firewall

Sl No	Parameters	SDN Firewall	Existing Firewall
1	Firewall Policy Management	POX Controller	Dedicated Firewall System
2	Network Protection Responsibility	All network elements	Dedicated network element
3	Firewall Rule Set	Access Control List	Flow based
4	Size of Firewall Rule Set	Random	Fixed
5	Firewall Rule generation	Dynamic	Static
6	Network Monitoring	POX Controller	Dedicated Firewall System
7	Failure Rate	6% to 8%	42%
8	Success Rate	96% to 99%	58%

E. Mitigating Ping Flood Attack using Packet Filter Firewall

In Fig. 11, X-axis represents flooding of infinite number of ping packets and Y-axis represents percentage of attack packets undetected. Approximately 6% to 8% of the ICMP Ping flood attack packets are not blocked by Packet Filter Firewall. When the ping flooding is detected by the packet filter firewall, flow rule is generated and installed in the flow table of the switch which is subjected to ping flood attack. This flow table entry completely blocks the Ping Flooding resulting in 100% packet drop as shown in Fig. 11. During the interval between attack detection and flow rule installation in flow table of a switch, some of the Ping flood packets are not blocked by the Packet Filter Firewall. This is the main reason for achieving 6% to 8% failure rate. Once the flow rule is installed, 100% packet drop is found and ping flood attack is successfully blocked by Packet Filter Firewall.

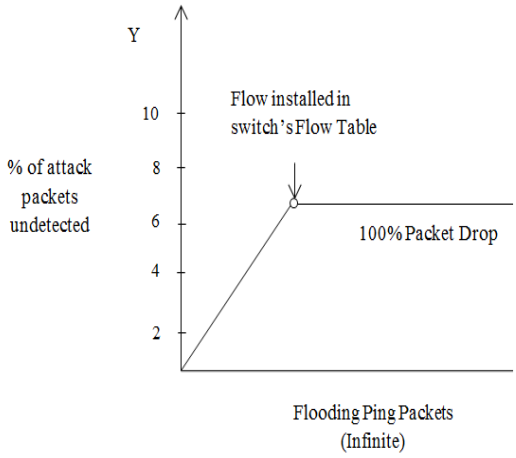

Fig. 11 Mitigating Ping Flood Attack

F. Mitigating Connection Flood Attack using Stateful Inspection Firewall

In Fig. 12, X-axis represents the SYN packets sent from client to sever for TCP connection and Y-axis represents the Number of successful TCP connections from a client to HTTP Web server establishment. In the above Fig., P1 means SYN Packet 1. This convention applies to rest of SYN packets sent. As shown in the Fig. 12, maximum of 2 continuous SYN packets can be sent from client at a time, i.e. 2 simultaneous TCP connections can exists at a time. Client has to wait for duration of 25 seconds in order to establish third TCP Connection to server. This scenario is shown in the Fig. 12, by making third SYN packet to wait before successful TCP Connection establishment. This technique is followed for rest of the SYN Packets from same client.

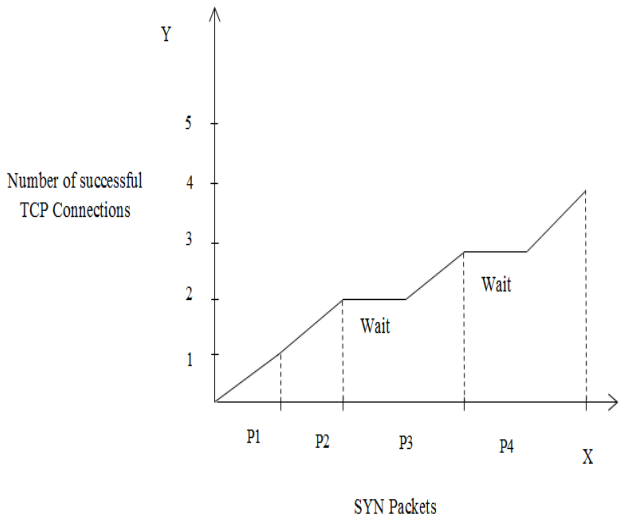

Fig. 12 Mitigating Connection Flood Attack

5. Conclusion

Dynamic and distributed firewall in SDN is implemented using 3 major techniques. The first and basic firewall technique is Packet Filter Firewall. This technique successfully able to overcome Ping Flood attack on 4 servers,

File server, Print server, Email server and Database server. As a result of this, server need not waste CPU cycles for processing unnecessary ping requests from clients. Server resources and bandwidth is made available for other respective services of the servers. Packet Filter Firewall achieves failure rate of 6% to 8% i.e. approximately 6% to 8% of the ICMP ECHO request ping packets are processed by the servers. Firewall fails to drop these ping packets successfully. Packet Filter Firewall achieves a success rate of 96% to 99%.

The second firewall technique implemented is Stateful Inspection Firewall. This firewall implemented overcomes the depletion of server resources by clients. It mitigates the SYN Connection Flood attack by the clients to HTTP web server. As a result, all the clients in the network are provided equal opportunity in consuming server resources efficiently. HTTP web requests from the clients get a response from web server within 25 seconds of time duration. HTTP responses by the HTTP server are in the HTTP requests order. In addition, HTTP server blocks the HTTP requests by the unknown hosts with the intimation to the POX controller about unknown host's request.

The third and final firewall technique implemented is Application Proxy Firewall. Application Proxy is developed in such a way that it successfully blocks access to the 2 servers namely, Facebook server and Youtube server and allows access to Google server. This firewall policy applies to all the clients in the network with Internet connectivity provided by NAT.

6. Future Work

The Proposed Dynamic and Distributed Firewall can be enhanced by integrating some of the potential features of Static Firewall to obtain Hybrid Firewall. Hybrid Firewall combines advantages of both dynamic and static firewall. Proposed Firewall design can be implemented in real time using actual controllers and switches and to prove SDN Firewall as a promising and effective solution. Proposed Firewall design can be implemented using other controllers such as NOX, RYU, FloodLight and OpenDayLight. Efficiency of firewall filtering is compared among the controllers.

References

[1] B. Khan, M. K. Khan, M. Mahmud and K. S. Alghathbar, "Security Analysis of Firewall Rule Sets in Computer Networks," *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on* , pp. 51,56, 18-25 July 2010.

[2] C. Pal, S. Veena, R. P. Rustagi and K. N. B. Murthy, "Implementation of simplified custom topology framework in Mininet," *Computer Aided System Engineering (APCASE), 2014 Asia-Pacific Conference on*, pp. 48-53, 10-12 Feb. 2014.

[3] D. Sethi, S. Narayana and S. Malik, "Abstractions for model checking SDN controllers," *Formal Methods in Computer-Aided Design (FMCAD), 2013*, pp. 145-148, 20-23 Oct. 2013.

[4] David Meyer, "The Software-Defined-Networking Research Group," *Internet Computing, IEEE* , vol.17, no.6, pp. 84-87, Nov.-Dec. 2013.

[5] E. W. Fulp, "Parallel Firewall Designs for High-Speed Networks," *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pp. 1-4, 23-29 April 2006.

[6] G. Paul, A. Pothnal, C. R. Mandal and B. B. Bhattacharya, "Design and implementation of packet filter firewall using Binary Decision Diagram," *Students' Technology Symposium (TechSym), 2011 IEEE* , pp. 17-22, 14-16 Jan. 2011.

[7] H. Hata, "A study of requirements for SDN switch platform," *Intelligent Signal Processing and Communications Systems (ISPACS), 2013 International Symposium on* , pp. 79-84, 12-15 Nov. 2013.

[8] Hyunmin Kim, Jaebeom Kim and Young-Bae Ko, "Developing a cost-effective OpenFlow testbed for small-scale Software Defined Networking," *Advanced Communication Technology (ICACT), 2014 16th International Conference on* , pp. 758-761, 16-19 Feb. 2014.

[9] I. Mothersole and M. J. Reed, "Optimising Rule Order for a Packet Filtering Firewall," *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pp. 1-6, 18-21 May 2011.

[10] Iman Kashefi, Maryam Kassiri and Ali Shahidinejad, "A Survey on Security Issues in Firewalls: A New Approach for Classifying Firewall Vulnerabilities," *International Journal of Engineering Research and Applications (IJERA)*, vol. 3, pp. 585-591, Mar. - Apr. 2013.

[11] J. Collings and Jun Liu, "An OpenFlow-Based Prototype of SDN-Oriented Stateful Hardware Firewalls," *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on*, pp.525-528, 21-24 Oct. 2014.

[12] K. Tantipongsakul and A. Khunkitti, "Dynamic Policy-Based Routing Using Firewall Rules," *Computer Modeling and Simulation, 2009. EMS '09. Third UKSim European Symposium on*, pp. 540-545, 25-27 Nov. 2009.

[13] L. Rodrigues Prete, C. M. Schweitzer, A. A. Shinoda and R. L. Santos de Oliveira, "Simulation in an SDN network scenario using the POX Controller," *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*, pp. 1-6, 4-6 June 2014.

[14] Liu Cong, Wang Wen and Wang Zhiying, "A configurable, programmable and software-defined network on chip," *Advanced Research and Technology in Industry Applications (WARTIA), 2014 IEEE Workshop on* , pp. 813-816, 29-30 Sept. 2014.

[15] M. Msahli, G. Pujolle, A. Serhrouchni, A. Fadlallah and F. Guenane, "Openflow and on demand networks," *Network of the Future (NOF), 2012 Third International Conference on the*, pp. 1-5, 21-23 Nov. 2012.

[16] M. Suh, Sae Hyong Park, Byungjoon Lee and Sunhee Yang, "Building firewall over the software-defined network controller," *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, pp. 744-748, 16-19 Feb. 2014.

[17] Mon-Yen Luo and Jun-Yi Chen, "Software Defined Networking across Distributed Datacenters over Cloud,"

Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on, vol. 1, pp. 615-622, 2-5 Dec. 2013.

[18] R. Jmal and L. Chaari Fourati, "Implementing shortest path routing mechanism using Openflow POX controller," *Networks, Computers and Communications, The 2014 International Symposium on* , pp. 1-6, 17-19 June 2014. 25

[19] R. L. S. Oliveir, A.A. Shinoda, C. M. Schweitzer and L. Rodrigues Prete, "Using Mininet for emulation and prototyping Software-Defined Networks," *Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on*, pp. 1-6, 4-6 June 2014. 26

[20] R. Trandafir, M. Carabas, R. Rughinis and N. Tapus, "FirewallPK: Security tool for centralized Access Control List management," *RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference, 2014*, pp. 1-6, 11-13 Sept. 2014. 27

[21] S. Khummanee, A. Khumseela and S. Puangpronpitag, "Towards a new design of firewall: Anomaly elimination and fast verifying of firewall rules," *Computer Science and Software Engineering (JCSSE), 2013 10th International Joint Conference on*, pp. 93-98, 29-31 May 2013. 29

[22] S. Zerrik, A. El ouadghiri, D. El ouadghiri, R. Atay, M. Bakhouya and J. Gaber, "Towards a decentralized and adaptive software-defined networking architecture," *Next Generation Networks and Services (NGNS), 2014 Fifth International Conference on*, pp. 326-329, 28-30 May 2014. 30

[23] Shie-Yuan Wang, "Comparison of SDN OpenFlow network simulator and emulators: EstiNet vs. Mininet," *Computers and Communication (ISCC), 2014 IEEE Symposium on*, pp.1-6, 23-26 June 2014. 31

[24] Shpetim Latifi, Arjan Durresi and Betim Cico, "Emulating enterprise network environments for fast transition to software-defined networking," *Embedded Computing (MECO), 2014 3rd Mediterranean Conference on* , pp. 294-297, 15-19 June 2014. 32

[25] T. Verwoerd and R. Hunt, "Policy and implementation of an adaptive firewall," *Networks, 2002. ICON 2002. 10th IEEE International Conference on*, pp. 434-439, 2002. 34

[26] William Stallings, "Firewalls," in *Cryptography and Network Security*, 5th ed. USA:Prentice-Hall,2010,ch. 22, sec. 3,pp.