International Journal of Engineering & Technology, 7 (2.24) (2018) 74-78

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Advanced firewall mechanism with OpenFlow in SDN

Ashutosh Phatak[1*], K. Vijayan[2], Ruturaj Kadikar[3], B. Amutha[4]

[1,2]Department of Information and Telecommunication Engineering
[3,4]Department of Computer Science Engineering
SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India
**Corresponding Author E-mail: ashutoshphatak0901@gmail.com*

Abstract

In recent years, penetration of Internet in the world is significantly increased due to technologies that enabled high speed broadband services, social networking and cloud based services. There is considerable increase in the number of users getting connected and hence large amount of user's vital data are flowing over Internet attracting serious threats and possible attacks from malicious users. To secure this free-flowing data, many security solutions have been presented, validated and implemented. But the majority of them are implemented with traditional networking techniques which itself is complex and hard to manage. This techniques primarily relies on manual configuration of devices which often results in policy conflicts that compromises network's security. This problem is addressed by Software Defined Networking, which breaks vertical integration by separating the control logic and data forwarding functionality, allowing flexible network architecture, network-wide visibility, simpler network management, etc. OpenFlow is the open standard that enables secure communication between controlling devices and data forwarding devices. In this paper, we propose and validate an approach to implement network-wide firewall in SDN by exploiting capabilities of OpenFlow standard to restrict flow of malicious and suspicious traffic flow in the network.

***Keywords**: Access Control List; Firewall; OpenFlow; REpresentational State Transfer; Software Defined Networking.*

1. Introduction

Internet has reached almost every house in the world and become a part of humans' basic needs. Every upcoming technologies, applications has a great usability of Internet and hence humans' dependency on it increases. Around 54.4% [17] of world population are Internet users and the number of users has increased thousand folds in last 18 years. A private network is a network within the specified user systems and servers where some restrictions are imposed to enable secured networking. Many government organizations, educational institutions, businesses are opting for having own private network since the number of IPv4 addresses are limited compared to the number of devices getting connected to public network like Internet. The main concern of these private networks is to protect information, documents, and databases of the organization which is solved by implementing security policies through firewall, proxy server, etc.

Presently these private networks are being implemented using traditional networking methods which itself has some loopholes. The security in the traditional network architecture primarily relies on manual configuration of the security solutions and the networking devices like router, switches. The network designers and deployment teams must use vendor specific commands to configure the technologies such as Firewall, Intrusion Detection System (IDS), IPSec [1] for implementing security policies. However, this manual configuration is more prone to configuration error, inter- and intra-domain policy conflicts resulting in security breaches. The centralized control in SDN encourages the enforcement of network-wide security policies and prevents policy collision.

Software Defined Networking (SDN) is the framework for network architectures that separate control logic of network from data forwarding plane [2] making the network management more straightforward. The control logic of the network is implemented in a logically centralised network controller making switching and routing devices as simple data forwarding devices as shown in Fig. 1. Famous organizations like Microsoft, Google, Yahoo Facebook, Verizon [2] has put interest in development of open standards for SDN. The OpenFlow is a protocol that enables communication between the control plane and the data plane. The controller uses OpenFlow protocol to pass switching, routing, load balancing or firewall policies onto data plane devices [10].

Firewall can be visualized as a security system based on predefined security rules used for monitoring and controlling incoming and outgoing packet traffic in a network. A typical firewall acts as a barrier between an internal trusted network and an external untrusted network such as the Internet. It is advantageous to implement firewall with SDN network architecture as the centralized control in SDN encourages the enforcement of network-wide security policies and prevents policy collision.

In this paper, a firewall security framework is proposed which is designed to provide network-wide security while inspecting incoming flows into the network. This solution gives the network administrator full control over security policy implementation and modification; simultaneously making the firewall immune to threats by monitoring network flows.

This paper is arranged as follows. Section 2 provides previous research on firewall. Section 3 presents a model of SDN based firewall framework, followed by Implementation in Section 4. Section 5 discusses validation and analysis of firewall. Conclusion and future work are covered in section 6.

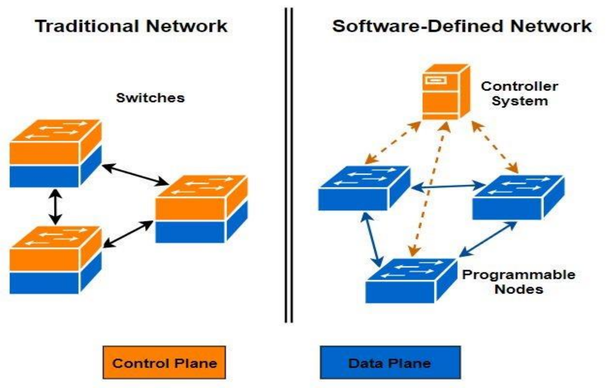


Fig. 1: Comparison between Traditional and SDN networks

2. Related Work

Security in SDN is vastly researched in recent years to exploit capabilities of SDN for enhancing network security. Since firewall is only device that operate at boundary of the network, many methodologies, ideas are presented on implementation of firewall policies. Hu et al.[3] highlighted possible security threats like insertion of false flow entries, spoofing controller, bypassing pre-defined policies, etc. The proposed security architecture for SDN emphasizes necessity of administrator authentication, enforcement of non-bypassing network policies, packet data scanning to ensure high level of security within the network.

Othman et al. [4] proposed the implementation of SDN firewall on POX controller with reactive approach in which a firewall module were running on controller and a learning switch module on OpenFlow switch. This learning switch module is made to trigger OpenFlow events on arrival of unknown packets and modify its flow table on instruction of firewall module. In reactive firewalling approach, Nife and Kotulski [5] came up with idea of possible reactive firewall mechanism using S-table and SecPolTable in addition to the flow table. Though the idea is for optimization of firewall performance in SDN networks, there is uncertainty over its implementation and validation. REFLO [6] is another firewall mechanism with reactive behaviour aiming to increase throughput. In REFLO, Visoottiviseth et al. designed the network topology that distribute the network traffic through multiple links having firewalls and a redundant link.

Zerkane et al [7] proposed a SDN firewall with proactive approach and included an Orchestrator at application plane that manages the security of the network. Here, Orchestrator controls many SDN controllers and is responsible for deployment of security policies in the network through all available controllers. Tran and Ahn [8] introduced topology discovery in a firewall concept called FlowTracker to improvise deployment of security policies by reducing addition of redundant entries in flow table.

DeCusatis and Mueller [9] used the concept of Virtualization of Firewall for Distributed Overlay Virtual Ethernet (DOVE) to secure communication between VMs and implemented using IBM 5000v as virtual switch with Juniper perimeter vSRX as virtual firewall. An application level firewall was developed by Shieha [10] using POX controller and configured to block all incoming traffic from Torrent and YouTube.

3. Firewall Framework

The SDN based network comprises of devices placed in three planes application, control and data - collaborated to achieve end-to-end network connectivity as shown in Fig. 2. The devices in data plane are simple data forwarding nodes and its role in network is interpreted by controller running at control plane. Various network services like switch, router, traffic monitoring, load balancing,

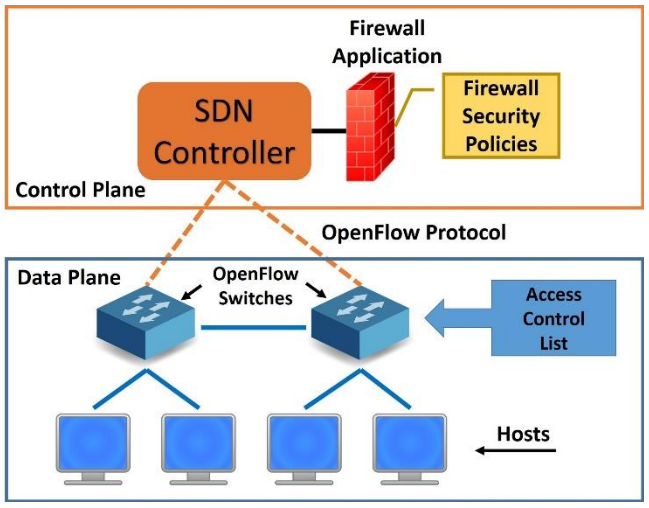


Fig. 2: Basic Firewall framework with OpenFlow in SDN

plane are simple data forwarding nodes and its role in network is interpreted by controller running at control plane. Various network services like switch, router, traffic monitoring, load balancing, firewall run in application plane on top of controller and define functionalities of network nodes, network policies, etc. A number of communication protocols exist between control plane and data plane but OpenFlow protocol is most popular and preferred open standard for communication. OpenFlow is evolved from version 1.0 to 1.5 and it launched with only 12 match fields and single flow table. The proposed firewall framework is based on latest version of OpenFlow i.e. ver 1.5 [11] which features 44 match fields and multiple flow tables (separate flow table for ingress and egress port) as future OpenFlow enabled switches will be governed by it.

The firewall framework is built on Ryu SDN controller and data plane nodes operating on OpenFlow ver 1.5 switch specifications. Ryu [12] is an open SDN framework for controller with modules and applications written in Python programming language and its architecture supports applications to be complied and run as it is part of the controller modules. Along with that, to operate as data forwarding device, Open vSwitch [13] is used which is an open source multilayer switch with licensed under open source Apache 2.0 license. The firewall rules are based on match fields specified in OpenFlow ver 1.5 and can be set to either allow or block the network flows depending upon header values such as MAC address, IP address (IPv4 or IPv6) and transport layer port number. The firewall application is running as module in Ryu controller and deploy firewall policies as flow entries into Open vSwitches. The application collects status of, whether connected or not, all available switches in network and accordingly the network administrator can set rules for every individual switch through a user interface of application. Additionally, the application continuously monitors policies installed in Open vSwitches to ensure it is not modified by any external or internal system and upon detection, the application re-route the flows of the network as preventive measure. To ensure scalability of mechanism to deploy over larger network, four Open vSwitches are used which connects four different user systems to each other and a controller to network.

Fig. 3 illustrates basic blocks of SDN firewall consisting firewall module, REST translation, list of switches and list of firewall rules. The heart of application is firewall module which co-ordinates with Ryu controller modules for implementing security rules in OpenvSwitches. The firewall policies can be accessed, set, deleted or modified on user interface through REST application interface (API). REST [14] is an acronym for REpresentational State Transfer and is an architectural style of web API that operates using four constraints (like HTTP commands) i.e. GET/POST/PUT/DELETE for reading, writing, modifying and deleting resource data. This constraints (or commands) , received from user interface, are decoded by REST translation module and accordingly firewall module takes suitable actions like enabling switch with firewall functionalities, maintaining list of connected switches, setting up new

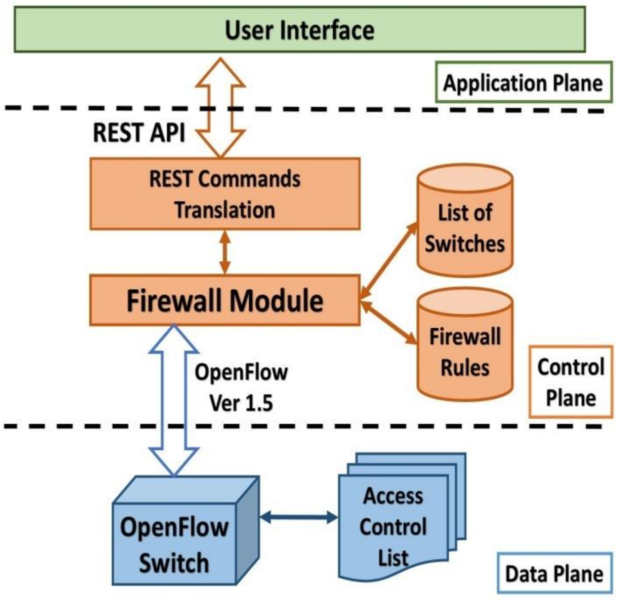


Fig. 3: Building blocks of SDN firewall application

rules, modifying or deleting existing rules, extracting flow entries from a switch, etc. The firewall module has two lists i.e. list of firewall rules and list of connected switches for seamless network-wide firewall policies deployment. The list of switches consist of datapath ids of all switches connected to controller and it assists the network administrator to implement switch specific firewall policies. This can prevent implementation of redundant rules in switches with setting different level of security at different part of the network (e.g. data center have very strict filtering whereas user network has basic network security). The list of firewall rules dictates pre-defined security policies governing access control in the network and it has different set of rules for every individual switches and VLANs. Every switch at data plane has access control list in their flow tables and this list is replica of rules defined for that switch in firewall application. The foundation of rules controlling access of various network traffic is on five fundamental parameters which are MAC address, IP address, Transport port number, type of connection and type of packets. The connection can be of TCP or UDP while transport port number identifies the requested service of network flow like HTTP, SNMP, FTP, Telnet, etc.

The access control list consist of 8 match fields of OpenFlow specification in which there are three pairs for MAC address (dl_sorc, dl_dest), IP address (ip4_sorc/ip6_sorc, ip4_dest/ip4_dest) and transport layer port numbers (tp_sorc, tp_dest) of source and destination. In remaining fields, dl proto field indicates whether the rule is for ARP, IPv4 or IPv6 packets whereas nw_proto field indicates network packets or transport layer connection. Latter field can be set for ICMP packets or TCP/UDP connections. Three pairs of fields combined with dl_proto and nw_proto determines type of network flow which can be either allowed or blocked to flow in network (e.g. nw_proto is set to TCP and tp_dst has port number 80, then the rule is defined for packet flows of HTTP). Any incoming packets for which there is no rule defined are dropped at switch and its header information is shared with firewall applicationby sending packet in message to controller. As soon as new rules for unknown trafficare defined, same are deployed in switches which are handling those network traffic.

4. Implementation

The test environment for SDN firewall was deployed in GNS3 network emulator application with five virtual machines (VMs), created in hypervisor, and three Open vSwitch devices. GNS3 [15] is an open-source application for emulation of complex network in simplified ways and support various legacy and open-source devices like switch, routers, security appliances, VMs, etc. One of these VMs has Ryu Framework installed and remaining VMs were used as user system. The system specifications of each component of test environment is given in Table 1.

Table 1: System specifications of Virtual Machines

System	Specifications	
SDN Controller	O.S.	: Ubuntu 16.04 LTS 64-bit
	Memory	: 4 GB
	Storage	: 50 GB
	Framework	: Ryu
User System	O.S.	: Bodhi Linux 64-bit
	Memory	: 512 MB
	Storage	: 10 GB

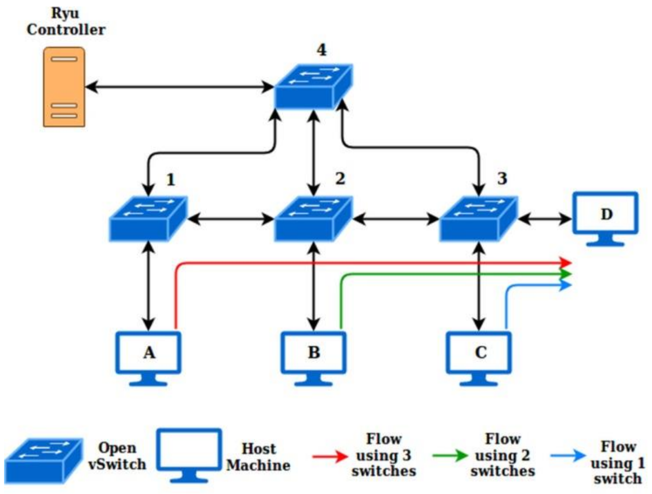
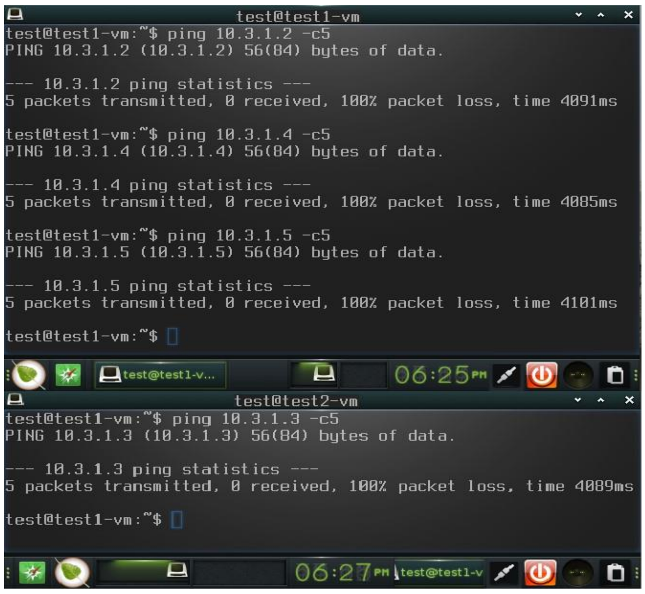


Fig. 4: Building blocks of SDN firewall application

Fig. 4 is the test environment and as shown in it, three user systems are connected to three separate Open vSwitch devices and all these are able to communicate through links between Open vSwitch. For evaluation purpose, fourth user system is used to collect network traffic generated by three user systems. Considering the limitation on having multiple physical network adapters on a system, an additional Open vSwitch is placed which connects the Ryu controller to all three switches with single physical network adapter. The test environment was split into three scenarios – three switches topology, two switches topology and single switch topology. To analyze the performance of these scenarios with firewall policies, the performance with three switches is compared with that of a single switch with same configuration. Each of these scenarios were tested by generating ICMP, TCP and UDP traffic from three user machines (VMs) destinated to fourth VM.

5. Validation and Results

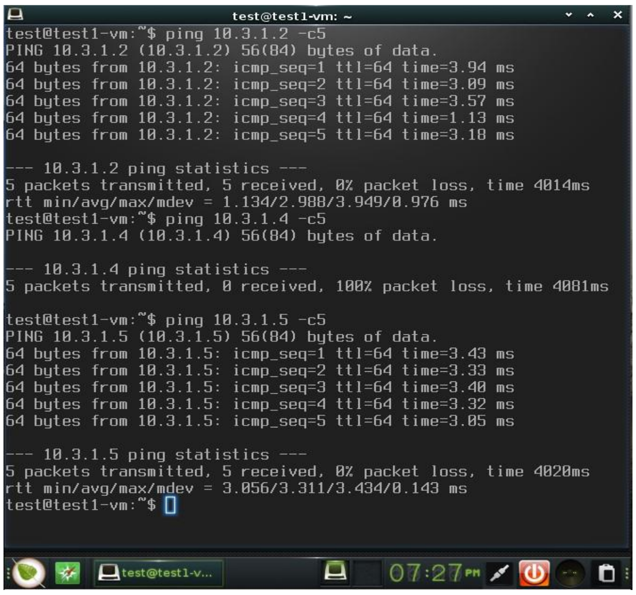
Distributed- Internet Traffic Generation (D-ITG) tool [16] is used for validation and evaluation of three scenarios by generating ICMP, TCP and UDP traffic. For validation, the firewall rules were set to block all TCP and ICMP packets and same were deployed in Open vSwitch devices as Access Control List (ACL). For TCP traffic, packets were generated using D-ITG tool and observed on Wireshark packet analyzer tool. For ICMP traffic, ICMP requests were generated using PING command on Command Line Interface (CLI) of three user systems and ICMP response were observed. Fig. 5,6,7 clearly shows, at beginning, Open vSwitch blocked all TCP and ICMP packets as dictated by rules stored in ACL. The responses were changed indicating flow of traffic through switch as the firewall rules were changed to allow these packets to flow. In case of ICMP packets, Open vSwitch 3 was later configured to allow all incoming ICMP traffic except traffic to or from Host 2 (IP



```
test@test1-vm
test@test1-vm:~$ ping 10.3.1.2 -c5
PING 10.3.1.2 (10.3.1.2) 56(84) bytes of data.

--- 10.3.1.2 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4091ms

test@test1-vm:~$ ping 10.3.1.4 -c5
PING 10.3.1.4 (10.3.1.4) 56(84) bytes of data.

--- 10.3.1.4 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4085ms

test@test1-vm:~$ ping 10.3.1.5 -c5
PING 10.3.1.5 (10.3.1.5) 56(84) bytes of data.

--- 10.3.1.5 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4101ms

test@test1-vm:~$
```

```
test@test2-vm
test@test1-vm:~$ ping 10.3.1.3 -c5
PING 10.3.1.3 (10.3.1.3) 56(84) bytes of data.

--- 10.3.1.3 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4089ms

test@test1-vm:~$
```

Fig. 5: ICMP packets before allowing through firewall rules



```
test@test1-vm: ~
test@test1-vm:~$ ping 10.3.1.2 -c5
PING 10.3.1.2 (10.3.1.2) 56(84) bytes of data.
64 bytes from 10.3.1.2: icmp_seq=1 ttl=64 time=3.94 ms
64 bytes from 10.3.1.2: icmp_seq=2 ttl=64 time=3.09 ms
64 bytes from 10.3.1.2: icmp_seq=3 ttl=64 time=3.57 ms
64 bytes from 10.3.1.2: icmp_seq=4 ttl=64 time=1.13 ms
64 bytes from 10.3.1.2: icmp_seq=5 ttl=64 time=3.18 ms

--- 10.3.1.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4014ms
rtt min/avg/max/mdev = 1.134/2.988/3.949/0.976 ms
test@test1-vm:~$ ping 10.3.1.4 -c5
PING 10.3.1.4 (10.3.1.4) 56(84) bytes of data.

--- 10.3.1.4 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4081ms

test@test1-vm:~$ ping 10.3.1.5 -c5
PING 10.3.1.5 (10.3.1.5) 56(84) bytes of data.
64 bytes from 10.3.1.5: icmp_seq=1 ttl=64 time=3.43 ms
64 bytes from 10.3.1.5: icmp_seq=2 ttl=64 time=3.33 ms
64 bytes from 10.3.1.5: icmp_seq=3 ttl=64 time=3.40 ms
64 bytes from 10.3.1.5: icmp_seq=4 ttl=64 time=3.32 ms
64 bytes from 10.3.1.5: icmp_seq=5 ttl=64 time=3.05 ms

--- 10.3.1.5 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4020ms
rtt min/avg/max/mdev = 3.056/3.311/3.434/0.143 ms
test@test1-vm:~$
```

Fig. 6: ICMP packets after allowing through firewall rules

address – 10.3.1.4) and same was validated using PING tool by generating ICMP traffic from Host 1 to Host 2 (IP address - 10.3.1.4) and other systems (Host 4 – 10.3.1.2, Host 3 – 10.3.1.5) as shown in Fig. 5 and 6.

The throughput for TCP traffic was evaluated to analyze network performance after allowing all incoming TCP connections in all three switches (Open vSwitch 2, 3 and 4). TCP connections were created between host 1, 2, 3 and 4 with 1 Mega Bytes of random data and packet size of 1500 Bytes, maximum permissible packet size in Ethernet standard, while the number of transmitting packets per second parameter was varied for every iteration from 500 to 2000 pps. At receiving end i.e. Host 4 system, number of received packets per seconds was recorded using D-ITG tool. Fig. 8 shows throughput observed in TCP connections established in 3 different scenarios and it is noticed that, for packet size of 1500 Bytes, network could not maintain throughput as number of transmitting packets per second is increased beyond 1250 pps. Another observation is that as the number of packets per second is increased,the relationship between the throughput and number of switches is observed to be inversely proportional i.e. as the number of switches increases, the throughput decreases. This is due to fact that each switch contributes a significant amount of delay that reduces the throughput.

In order to observe response to UDP traffic, multiple simultaneous flows were generated with each transmits 1 Mega Bytes of randomdata and packet size set to 1500 Bytes, while the number of simultaneous flows varied from 10 to 200 flows. It was observed that the host system could not handle the processing of simulation when the number of flows were increased beyond 200. Like forUDP traffic, multiple ICMP flows were generated with 64 Kilo Bytes of random data and average delay was observed in all three scenarios.

Fig. 9 depicts the relation between number of flows andaverage jitter observed in all three scenarios. It is evident from figure that, the average jitter (in msec) increased rapidly as the number of flows increased, but at higher number of flows, the average jitter increased gradually. Thus the graph resembles an inverse exponential curve indicating the network was able to manage buffering of UDP traffic. It was noted that there was marginally small amount of increase observed as the number of switches increased from one to three. The maximum average jitter observed for this setup was limited to 2 msec.

Fig. 10 describes the relation of delay in ICMP traffic with the number of simultaneous flows through different number of switches. It was observed that as the number of flows increased, the amount of

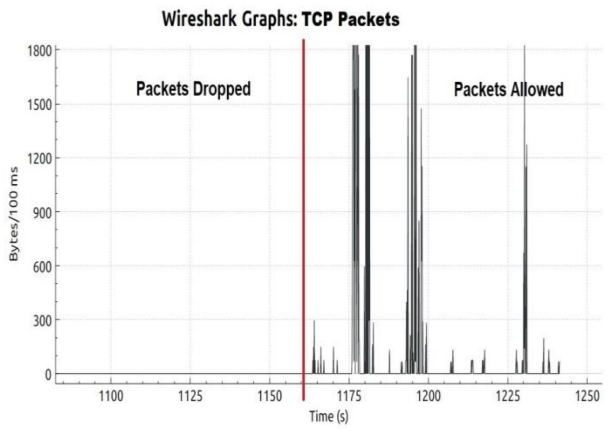


Fig. 7: TCP traffic – before & after allowing through switches

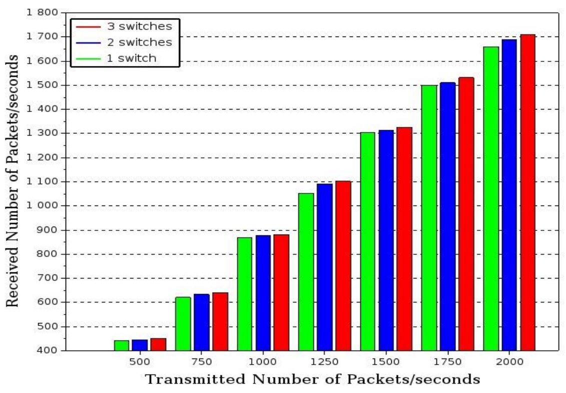


Fig. 8: Throughput analysis of TCP connection

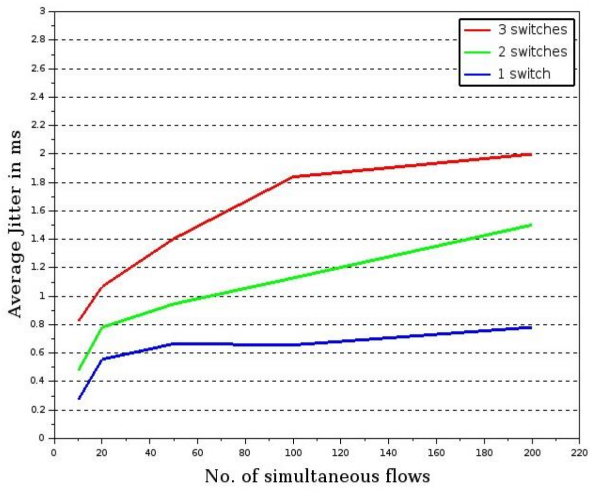


Fig. 9: Average jitter in UDP traffic

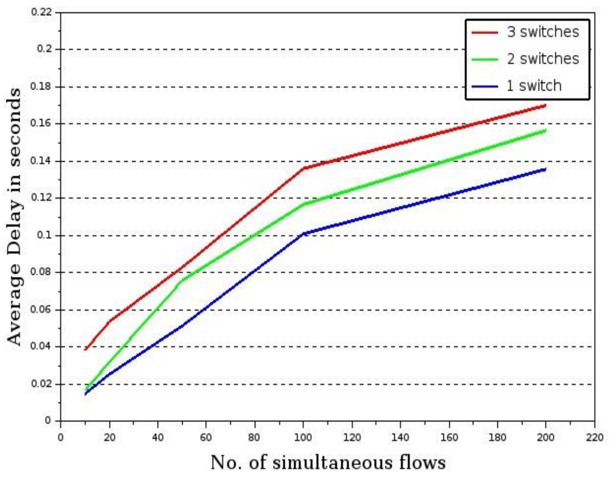


Fig. 10: Average delay in ICMP packets

delay increased indicating presence of congestion in the network. Also it was noticed that the amount of average delay produced by three switches was not cumulative average delay produced by individual switches, instead the delay with three switches was slightly higher than that with single switch.

6. Conclusion

The firewall mechanism is tested on a prototype network in three different scenarios for three different packets namely, ICMP, TCP and UDP and it shows that SDN based firewalls with OpenFlow can be promising method for defending malicious threats in scalable networks. SDN features flexible network policing and network device programmability while OpenFlow protocol provides MAC/IP/TCP layer traffic filtering in simple data forwarding device. This helps the implementation of network-wide security polices with maintaining performance of network similar to that of traditional networks. The proposed firewall is validated on network emulation platform and implementing it with OpenFlow v1.5 ensures to incorporate future versions of OpenFlow to enhance network security. Further, the firewall application can include other security features like deep packet inspection, intrusion detection for better security prospects.

Acknowledgement

We would like to express our gratitude to the research group, Software Defined Research Lab, Department of Computer Science Engineering, SRM Institute of Science and Technology for their kind assistance and providing us with the required resources.

References

[1] I. Ahmad, S. Namaly, M. Ylianttilaz and A. Gurtov, Security in Software Defined Networks: A Survey, IEEE Communications Surveys & Tutorials , Volume: 17, Issue: 4,(August 2015), pp 2317 - 2346.

[2] D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, Software-Defined Networking: A Comprehensive Survey, Proceedings of the IEEE, (2014), 14-76.

[3] Z. HU, M. WANG, X. YAN, Y. YIN and Z. LUO, A Comprehensive Security Architecture for SDN, 18th International Conference on Intelligence in Next Generation Networks, (February 2015), pp 30-35.

[4] W. M. Othman, H. Chen, A. Al-moalmi and A. N. Hadi, Implementation and performance analysis of SDN firewall on POX controller,IEEE 9th International Conference on Communication Software and Networks (ICCSN), (2017), pp 1461-1466.

[5] F. Nife and Z. Kotulski, Multi-level Stateful Firewall Mechanism for Software Defined Networks, Springer International Publishing, (June 2017), pp 271–286.

[6] V. Visoottiviseth, S. Lertviriyasawat, P. Suppiyatrakoon, P. Chitkornkitsil and N. Yamai, REFLO: Reactive Firewall System with OpenFlow and Flow Monitoring System, Proceedings of the 2017 IEEE Region 10 Conference (TENCON), Malaysia, (December 2017), pp 2273- 2278.

[7] S. Zerkane, D. Espes, P. Le Parc, and F. Cuppens, A Proactive Stateful Firewall for Software Defined Networking, Springer International Publishing, (March 2017), pp 123-138.

[8] T. V. Tran and H. Ahn, Flowtracker: A SDN Stateful Firewall Solution with Adaptive Connection Tracking and Minimized Controller Processing, International Conference on Software Networking (ICSN), (May 2016).

[9] C. DeCusatis and P. Mueller, Virtual Firewall Performance as a Waypoint on a Software Defined Overlay Network, IEEE 6th International Symposium on Cyberspace Safety and Security (CSS), (August 2014).

[10] A. Shieha, Application Layer Firewall Using OpenFlow, Interdisciplinary Telecommunications Graduate Theses & Dissertations, Paper 1, (2014).

[11] Open Networking Foundation, OpenFlow Switch Specification - Version 1.5.0, (December 19, 2014).

[12] Ryu Developing Team, Ryu Documentation, Release 4.21, (January 19, 2018).

[13] What is Open vSwitch? [Online] – Open vSwitch www.openvswitch.org

[14] What Is REST? [Online] – REST Tutorial www.restapitutorial.com/lessons/whatisrest.html

[15] Getting Started with GNS3 - docs.gns3.com/1PvtRW5eAb8RJZ11ma EYD9 aLY8kkdhgaMB0wPCz8a38/index.html

[16] A. Botta, A. Dainotti and A. Pescape, A tool for the generation of real-` istic network workload for emerging networking scenarios, Computer Networks (Elsevier), Volume 56, Issue 15, (2012), pp 3531-3547.

[17] INTERNET Usage Statistics - www.internetworldstats.com/stats.htm

[18] S.V.Manikanthan and T.Padmapriya "Recent Trends In M2m Communications In 4g Networks And Evolution Towards 5g", International Journal of Pure and Applied Mathematics, ISSN NO: 1314-3395, Vol-115, Issue -8, Sep 2017.

[19] S.V. Manikanthan, T. Padmapriya "An enhanced distributed evolved node-b architecture in 5G tele-communications network" International Journal of Engineering & Technology (UAE), Vol 7 Issues No (2.8) (2018) 248-254.March2018.

[20] S.V. Manikanthan, T. Padmapriya, Relay Based Architecture For Energy Perceptive For Mobile Adhoc Networks, Advances and Applications in Mathematical Sciences, Volume 17, Issue 1, November 2017, Pages 165-179