Centralized Firewall for Software-Defined Networking (SDN)

Sheetal Khodbhaya[1], Nimit Tiwari[2], Sachin Mahto[3], Jishnu Unnikrishnan[4],

Prof. K.S. Charumathi[5]

[1,2]Student, Department. of Computer Engineering, Pillai College of Engineering, Maharashtra, India
[3,4]Student, Department of Computer Engineering, Pillai College of Engineering, Maharashtra, India
[5]Professor, Department of Computer Engineering, Pillai College of Engineering, Maharashtra, India

---***---

Abstract - *Software-Defined Networking (SDN) is an architecture that aims to make networks agile and flexible by introducing programmability in networks. In SDN, the functionality of the network device is divided into control plane and forwarding plane. At the control plane, the SDN Controller provides APIs which can be used by the applications. It gives the forwarding logic in the form of flow rules to the OpenFlow Switches, at the forwarding plane, to forward the packets. We propose to implement a centralized firewall for SDN which will get the network details from the SDN Controller, analyze it, and will push the rules to the SDN controller using RESTful APIs.*

KEYWORDS: Software Defined Networks (SDN), Firewall, Openflow Protocol, OpenFlow-Enabled Switches, SDN Controllers.

1. INTRODUCTION

In traditional networks, both control and data planes are tightly integrated in physical devices. To specify routing policies in traditional networks, network administrators must maintain forwarding rules individually in all switches and routers in the network. In contrast, SDN has brought significant changes to how networks function by decoupling the control plane from the data plane. The decoupling abstracts the higher level functionality and moves the intelligence of network configuration to a centralized controller. This innovation has influenced both industries and academic institutions to persistently work towards adaptation and evolution of SDN. The two main advantages of SDN (central programmability and visibility) tremendously improve cost-effectiveness and ease of maintenance in these complex networks.

1.1 Definitions

Software-Defined Network (SDN) : It is an architecture that acts as a strategic control point created to address the issues related to integrated networks. SDN is not a mechanism, it is a framework to solve a set of issues related to network and packets. To increase the cost-effectiveness and logical control, SDN provides services like flow paths, packet handling and topology. SDN enhances open interfaces and programmability of the network through separation of network traffic delivery and network configuration. Despite the hype, SDN controllers and related protocols are rapidly evolving the current network technologies to address the demands for scaling in complex enterprise networks. The rate at which SDN frameworks are evolving continues to overtake attempts to address their security issues. Software defined networking enables the network creation without any use of hardware, hence it is a cost saving work. SDN was commonly connected with the OpenFlow protocols for remote communication with the network plane of network switches.

Firewall: It is a system designed to prevent unauthorized incoming network packets, which come from various sources, as well as outgoing network packets. It can monitor and control the flow of data which comes into the network from different sources, and works on the basis of predefined rules. Firewalls generally keep a barrier between a restriction, secure internal network and other outside networks, such as the Internet, that is presumed not to be safe or reliable. They can be characterized as either hardware or software firewalls.

RESTful API's: They are used for communication between the SDN and firewall. REST stands for REpresentational State Transfer. It means when a RESTful API is called, the server will transfer to the Application (client) a representation of the requested resource. RESTful API's Uses HTTP requests such as GET, PUT, HEAD, POST and DELETE data. They use GET to retrieve a resource; PUT to change the state of or update a resource, POST to create that resource; and DELETE to remove it. Binding to a service through an API is a matter of controlling how the URLs are decoded which can be used for communication between the firewall and SDN controller.

OpenFlow : It is a communication protocol that allows a server to tell networks switches where to send packets. The OpenFlow protocol is used to differentiate between an OpenFlow controller and OpenFlow switches. An

OpenFlow controller instructs the OpenFlow switch on how to manage incoming data packets and open interfaces for remotely controlling the forwarding tables in network switches, routers, and access points. OpenFlow is not an SDN and it is not the only protocol which can be used for communication within a software defined networking environment. However in current condition OpenFlow has been standardized and most widely used in these applications. OpenFlow is an open API. It has many similarities to the x86 instruction set for the networks. As such OpenFlow for software defined networks provides an open interface to networking nodes including routers, switches and the like. It enables visibility and openness in the network. The OpenFlow Switch data flow consists of a Flow Table. The OpenFlow controller maintains the communication channels to the OpenFlow switches and maintains the local state graph of the switches and exposes the RESTful API's to the Applications. SDN OpenFlow is based on an Ethernet switch, with an internal flow-table, and a standardized interface to add and remove flow entries.

2. SYSTEM ARCHITECTURE

The system architecture is given in Figure 3.1. Every block present is explained in this section

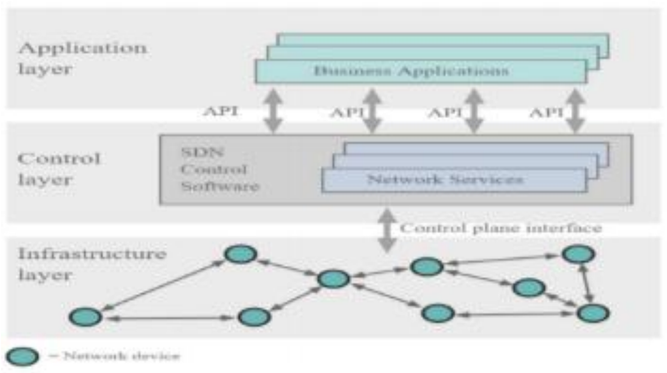


Fig 2.1 System Architecture of SDN

The three major parts of the frame of SDN are switch, controller and the interface needed for communication.

2.1 Switch

Switches are taken as hardware that can be operated via open interface. An Open-Flow switch has three parts, flow table, set of command and secure channel. The flow table stores the data regarding the flow entry for packets and lookup and forwarding. These flow entries consist of match fields, counters and sets of instructions that handle the matching of packets. On arrival of the packets at the switch, the packet header is being extracted and matched with the matching fields. When a matching entry is found, an appropriate set of instructions is applied and in case of a failed match the action will be taken according to the table miss flow entry. For eg. Dropping packets, continue the match process on the next flow table.

2.2 Controller

The control plane is the main part of the SDN architecture, so it is very important to give exact concern towards the design parameter of the controller. Controller provides a programming interface to the network. Multiple controllers are being used to hold the backup of data of the controller that controls the whole network. According to the experiments performed, the controller Open Network Operating System has good performance on clusters, linkup and throughput whereas Open-Daylight works well with topology discovery and stability. A controller is designed in such a way that it can handle upto 6.000.000 flows/sec5.

2.3 Interface for communication

Various protocols are used to communicate between the control plane and the data plane. RESTful API's are used for communication between the Firewall application and SDN Controller.

3. APPROACH

Centralized approach is being used in the SDN to solve the problem that is observed in traditional networks. It provides the programmable platform to design the network. The SDN architecture differs from other local architecture in terms of the carrier grade network. The separation of the plane makes it quite easy to implement new protocols and applications. The three major components of the frame of SDN are switch, controller and the interface needed for communication and packet transfer.

There were two approaches considered in implementing firewall: a) pre-installing the rules onto the switch's flow table and b) handling the packets directly as they come in. We chose to handle the incoming packets directly because of the flexibility in management. One downside of this method is that too many packets can be delivered to the controller and take up a large portion of its resources; it is a lot more efficient to block unnecessary packets at the switch level. To cope with this issue, the user can also decide to install a 'deny' flow modification on the switch to continue dropping similar packets for a certain time period. The logic of this firewall is as follows: each packet headers are checked against the firewall rule from highest to lowest priority, and performs specified action once matching fields are found in the rule. Any unmatched packets are dropped. Installing firewall rules are possible from an external entity through a text-based user interface.

4. IMPLEMENTATION

The implementation details are given in this section as, there were two tactics considered in implementing the firewall, In order to test the workability of this firewall, the following programs were used:
Virtual Box - offers a background for virtual networks to be formed.
Mininet - provides virtual SDN network topology.
POX - SDN controller.

The key technique used within software defined networking is to structure the network architecture so that the application layer, control layer and the infrastructure layers are separated and individually definable. Finally a simulated network is built on a mininet network simulator and random network traffic is generated from hosts to the servers. The firewall is able to identify any suspicious activity & alert the concerned parties.

4.1 Algorithm / Methodology/Techniques

We will create an SDN network for our project using an emulation tool called Mininet. So first we downloaded the oracle virtual box in windows 7 or 10. Then we downloaded the mininet tool and imported into the virtual box to run it. We configured the network settings of the mininet according to our project and then we ran we will create an SDN network for our project using an emulation tool called Mininet. So first we downloaded the code for the pox controller. Miniedit is used which was run to show the network topology consisting of hosts and switches and also the controller. A dialog box in miniedit can be used to specify the ip address of hosts that we want to connect to or block the host from connecting to. The pox console and the mininet can be connected and hence we can efficiently monitor the traffic flow and also see it coming from different hosts. A virtual switch will be used in the mininet for connection between the host and the controller.

4.2 SDN Firewall Commands :

[A] Add:
This command takes in the parameters and adds them to the list of rules in a dictionary form. This command has following functions:
• If a repeated name appears, it warns the user and doesn't add.
• The rule list is in the order of priority, and users can decide priority for each rule. If the user doesn't specify a priority, it is added to the end of the list.
• Timeout is an integer in units of seconds.
• Priority is optional. The highest priority that users can assign is one. When the indicated priority is already taken, the new rule gets placed at that priority and the rest is pushed back one priority. This command has following syntax: input in the form of (name, match, action, priority (optional)
• Anything non-integer is added with the quotation marks.
• When putting multiple match fields, all the match pair need to be put inside a parenthesis
• Four options for the action field are: 'allow', 'deny', ('deny', timeout): when idle & hard have equal timeout. This installs the rule to switch that rejects all similar packets for the time specified. ('deny', idle timeout, hard timeout).
[B] Delete:
This command takes the rule name as an input, and deletes from the rule. This command has following functions:
• If the indicated name doesn't exist, it warns the user.
• The name is case-sensitive, and does not need to be in quotation marks.

[C] Show:
These commands show the name of the rules. This command has following functions:
• The names of the rules are listed in the order of priority this command has following syntax:
• No additional input necessary.
[D] ShowComplete:
This command Shows complete entry of the rules. This command has following functions:
• List each rule in the order of priority, name, match, allow this command has following syntax:
• No additional input necessary.

[E] SwitchPro:
This command takes name and new priority as input, and switches list order. This command has following functions:
• The indicated rule gets placed at the newly assigned priority, and the rest of the rules gets pushed back one.

• Input should be in order of name, priority separated by a comma.
• Name does not need to be in quotation marks.

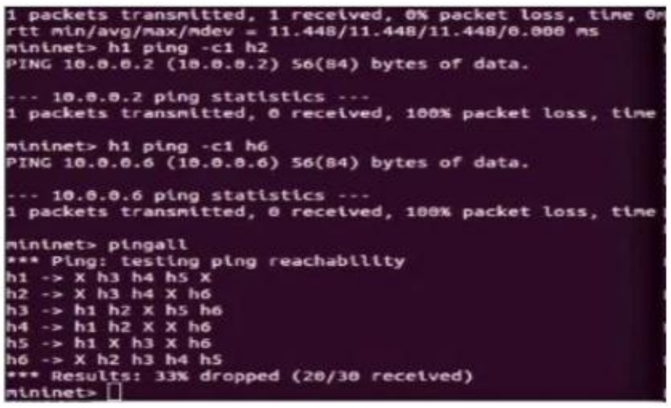
[F] Set Timeout:
This command sets a general timeout to drop similar packets in the future. This command has following functions:
• Checks that inputs are valid integers.
• Input in the order of idle timeout, hard timeout separated by a comma.

5. EXPERIMENTAL RESULTS

Mininet is a network emulator that enables the creation of a network of virtual hosts, switches, controllers, and links. Mininet creates a network with hosts. It runs on one terminal. In another terminal, the POX controller is executed, which works as the firewall and filters the traffic as per the rules specified in it.

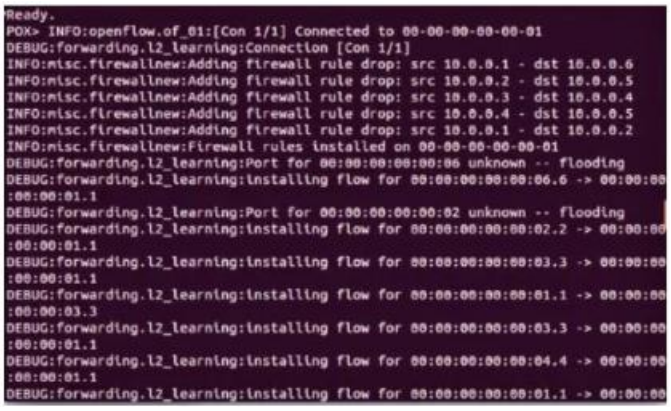
An OpenFlow-enabled switch runs at the gateway connecting a network under protection and the rest of the network. The firewall controller can be potentially hosted anywhere in the network. The security rules are specified in the flow table which is maintained in both the OpenFlow-enabled switch and the firewall controller. An entry in the flow table specifies the security rule for handing a traffic flow.

The switch acts as a simple packet-pusher based on the security rules defined in its flow table. The firewall controller makes use of its flow table to keep track of the control decisions on traffic flows. A designated communication channel is maintained between the switch and the firewall controller. Through this channel, the switch sends the information about the unidentified traffic flows to the controller for inspection, and the controller sends control decisions to the switch. When the switch cannot match a packet to a rule in its flow table, it sends the packet to the controller for inspection. After inspection, the controller informs the switch about its control action on a flow, and it also memorizes the control decisions in its flow table. The switch can also update the controller about the statistics. First, on the setup phase of the firewall, the controller will send OpenFlow commands to install firewall rules. Each firewall rule is a flow entry consisting of source MAC address matching field, destination MAC address matching field, and the action set to DROP action. After the setup phase, every packet going through the switches will be matched against flow table entries (including the firewall rules). If the packet matches one of the firewall rules (same source and destination MAC addresses), the switch will drop the packet immediately. When the switch cannot match a packet to a rule in its flow table, it sends the packet to the controller for inspection. After inspection, the controller informs the switch about its control action on a flow, and it also memorizes the control decisions in its flow table. The switch can also update the controller about the statistics. The rule processing module translates the rule specification to a collection of rule data structures, matches traffic flows to rules, and maintains the structure of the flow table. The configuration module translates the user-specified configuration into a data structure that is recognizable by the core module. By this flow table, traffic will be reduced and input flow will reduce the DDoS attack Therefore, every hardware device in the network will behave as a firewall and filter out unwanted traffic. It can be said that the flow concept is the backbone of OpenFlow and SDN.



```
1 packets transmitted, 1 received, 0% packet loss, time 0
rtt min/avg/max/mdev = 11.448/11.448/11.448/0.000 ms
mininet> h1 ping -c1 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.

--- 10.0.0.2 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time

mininet> h1 ping -c1 h6
PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.

--- 10.0.0.6 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time

mininet> pingall
*** Ping: testing ping reachability
h1 -> X h3 h4 h5 X
h2 -> X h3 h4 X h6
h3 -> h1 h2 X h5 h6
h4 -> h1 h2 X X h6
h5 -> h1 X h3 X h6
h6 -> X h2 h3 h4 h5
*** Results: 33% dropped (20/30 received)
mininet> 
```

Fig 5.1 Packet dropped by Firewall in SDN



```
Ready.
POX> INFO:openflow.of_01:[Con 1/1] Connected to 00-00-00-00-00-01
DEBUG:forwarding.l2_learning:Connection [Con 1/1]
INFO:misc.firewallnew:Adding firewall rule drop: src 10.0.0.1 - dst 10.0.0.6
INFO:misc.firewallnew:Adding firewall rule drop: src 10.0.0.2 - dst 10.0.0.5
INFO:misc.firewallnew:Adding firewall rule drop: src 10.0.0.3 - dst 10.0.0.4
INFO:misc.firewallnew:Adding firewall rule drop: src 10.0.0.4 - dst 10.0.0.5
INFO:misc.firewallnew:Adding firewall rule drop: src 10.0.0.1 - dst 10.0.0.2
INFO:misc.firewallnew:Firewall rules installed on 00-00-00-00-00-01
DEBUG:forwarding.l2_learning:Port for 00:00:00:00:00:06 unknown -- flooding
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:06.6 -> 00:00:00
:00:00:01.1
DEBUG:forwarding.l2_learning:Port for 00:00:00:00:00:02 unknown -- flooding
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:02.2 -> 00:00:00
:00:00:01.1
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:03.3 -> 00:00:00
:00:00:01.1
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:01.1 -> 00:00:00
:00:00:03.3
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:03.3 -> 00:00:00
:00:00:01.1
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:04.4 -> 00:00:00
:00:00:01.1
DEBUG:forwarding.l2_learning:installing flow for 00:00:00:00:00:01.1 -> 00:00:00
```

Fig 5.2 Execution of Firewall Rules

In the above figures, the firewall's rules are tested with the pingall command, which runs on one terminal to check the connectivity with all the hosts of the network. The result indicates that one host cannot connect with the other, which can be added as a drop packet rule in the firewall.

6. Requirement Analysis

The Requirement details are given in the section given below:

6.1 Software

The experiment setup is carried out on a computer system which has the following software specifications as given in Table 5.1

Table 5.1 Software Requirements

Operating System	Windows 10 with Oracle Virtual Box
Programming Language	Python

6.2 Hardware

The experiment setup is carried out on a computer system which has the following hardware specifications as given in Table 5.2

Table 5.2 Hardware Requirements

Processor	2 GHz Intel
HDD	180 GB
RAM	Minimum 4 GB

7. CONCLUSION

The popularity of SDN in the IT world today is demonstrated by the numerous startups that proliferate in this area. Though there exist outstanding firewalls in the market, it is quite costly for companies to install numerous firewall hardware across the entire network to preserve high security. Also as mentioned earlier, replacing a firewall is a serious pain – from physically replacing the firewall, reconfiguring each device related to the firewall to troubleshooting. Besides the inconvenience, one wrong turn can put the entire network at risk. Thus, SDN is not only revolutionary in making the control flexible and manageable, but also for firewalls to achieve programmability by separating the firewall hardware and the control software. An OpenFlow-based firewall with a straightforward UI that integrates priority switching can bring another wave of innovation in the Internet world.

8. FUTURE SCOPE

The main goal is to draw some reasonable answers. Can we (and how to) leverage the new features provided by SDN to enhance network security?. Based on research paper and in-depth analysis of SDN features and their applications discussed, we claim that SDN can clearly enhance network security functions in the following points. First, its ability of controlling network flows dynamically can provide more flexible deployments of security functions on a network because it allows us to enable security functions on SDN-enabled network devices without installing additional devices (e.g., middleboxes). Second, its network-wide visibility can realize network-wide monitoring in terms of security. This ability provides a holistic view to us, and thus we can comprehend network attacks widely distributed in the Internet (e.g., network-wide scanning or DDoS) much more efficiently than legacy network monitoring systems. Third, its programmability helps us develop more advanced network security functions. We can (relatively) easily implement a prototype security system without putting much effort. As such, SDN features can be leveraged in accelerating the development of new and advanced network security functions. We will introduce the SDN technology and systematically and investigate its usage for security.

Currently, this design only looks at the packet header fields to determine the action. Perhaps, future developers can further improve this logic by incorporating SDN capacities to improve security by observing the entire network flow and efficiently block the network attacks in the early stage without having to perform deep packet inspection.

9. ACKNOWLEDGEMENT

It is our privilege to express our sincerest regards to our supervisor Prof. K.S.Charumati for the valuable inputs, able guidance, encouragement, whole-hearted cooperation and constructive criticism throughout the duration of this work. We deeply express our sincere thanks to our Head of the Department Dr. Sharvari Govilkar and our Principal Dr. Sandeep M. Joshi for encouraging and allowing us to present this work.

REFERENCES

1. M. Suh, S. H. Park, B. Lee, and S. Yang, "Building firewall over the software-defined network controller," in Proceedings of the 16th International Conference on Advanced Communication Technology: Content Centric
2. Network Innovation!, ICACT 2014,pp. 744–748, Republic of Korea, February 2014.
3. J.G.V. Pena andW.E.Yu, "Development of a distributed firewall using software defined networking technology," in Proceedings of the 4th IEEE International Conference on Information Science and Technology, ICIST 2014, pp. 449–452, China, April 2014.
4. T. Javid, T. Riaz, and A. Rasheed, "Layer 2 firewall for software defined network," in Proceedings of the Conference on Information Assurance and Cyber Security, CIACS 2014, pp. 39–42, Pakistan, June 2014.
5. K. Kaur, K. Kumar, J. Singh, and N. S. Ghumman, "Programmable firewall using Software Defined Networking," in Proceedings of the 2nd International Conference on Computing for Sustainable Global Development,
6. INDIACom 2015, pp. 2125–2129, India, March 2015.
7. T. V. Tran and H. Ahn, "A network topology-aware selectively distributed firewall control in sdn," in Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), pp. 89–94, IEEE, Oct 2015.
8. B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: past, present, and future of programmable networks," IEEE Communications Surveys & Tutorials, vol. 16, no. 3, pp. 1617–1634, 2014.
9. H. Farhady, H. Lee, and A. Nakao, "Software-defined networking:A survey," Computer Networks, vol. 81, pp. 79–95, 2015.
10. opennetworking.org website
11. www.brianlinkletter.com website
12. Opensourceforu.com website
13. openflow.stanford.ed