**IOSR Journal of Engineering (IOSRJEN)** **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 08, Issue 6 (June. 2018), ||V (III) || PP 24-30

# SDN Based Firewall

Viren Wadhwani, Yash Bajaj, Yogesh Rohra
*Computer Department Vivekanand Institute of Technology*

**ABSTRACT:** In today's evolving technology, the need to adapt is the only necessity. The same is applicable when we are talking about the filed of Network and Securities i.e the SDN technology. SDN is the newest evolving topic in the filed of Networking. The are many problems faced by the traditional networks. These are solved by SDN. It is a way of making networks more manageable and more dynamic. It is very helpful in realizing the whole network beforehand thus giving a clear idea of the topologies required and the issues which can be solved even before setting the network up. But as in a new technology comes up it brings in new threats and issues. Thus our aim is to build a firewall over this new technology to protect the integrity of it. In this paper we focus on designing and developing a Firewall for our SDN controller which in turn performs the function of safe guarding the network from inside as well as the outside internet.To perform experiment, we have installed Mininet emulator for creating network topologies. In this paper, we present the implementation details of the firewall application.

Date of Submission: 01-06-2018 Date of acceptance: 16-06-2018

## I. INTRODUCTION

The current internet network is well set and huge in terms of topologies and its connections all over the world. But it has its drawbacks too. The data and the control plane being together yet makes it difficult in finding optimality in terms of routing, maintaining security and much more. Also adding or removing even a single router makes us change the whole network and is tedious. In a view of finding solution the SDN technology was introduced in 2005 where in the control plane gets totally removed and what we get is a controller i.e. the brain of the network and a bunch of routers that are meant for handling data and routing them. The new brain so formed is called as the SDN controller in the world of SDN. A controller responsible for taking the routing, protocol decisions whenever a packet arrives at its door.

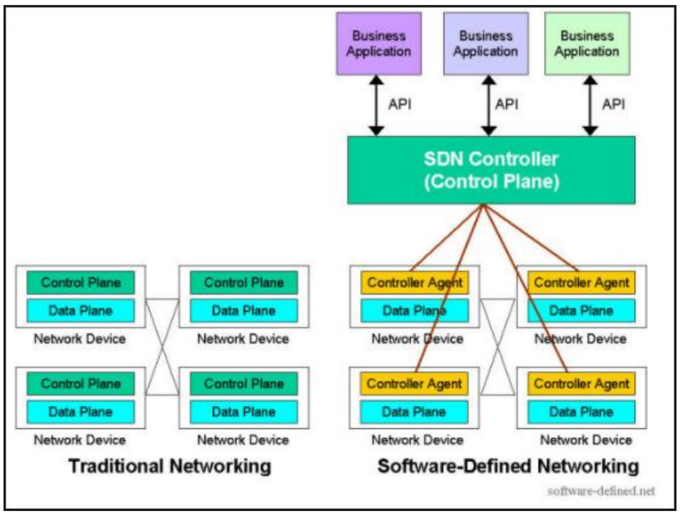
Our project aims at building a firewall over this controller that will in turn help the controller to take the decisions regarding the packets that arrive at the controller. The controller accompanied with the firewall will have the ability to accept, drop, or reject the incoming packets thus ensuring the safety of the system from malicious packets. The Firewall thus created will stand in between the controller and outside internet. Also it has the ability to handle the flow of packets shown by the controller thus providing an all rounder functionality where the controller is the mega brain and has the record of every bit of data irrespective whether it flows in our out of network. Also the firewall which is coded in Python, provides a simplified GUI for the users to set up the firewall as per his/her convenience and also it will be able to save the system from any malicious attack by an intended attacker.

Thus the project aims at emulate a network topology. The implementation includes the respective emulation of the network along with the firewall.

The remainder of this paper is organized as follows, Section 2 presents the comparison of this project with the existing system, followed by Section 3 which states the proposed system and the implementation of each block of the system, the Conclusion is shown in section 4 and in the last section the references are presented.

## II. COMPARISON WITH EXISTING SYSTEM

On viewing the existing system and comparing it with the project, we find that the basic disadvantages are the more privileged uses of the SDN technology.
As mentioned earlier, the existing networks are highly complex. The actual comparison between the two can be depicted by the figure above.



The traditional and yet existing system consists of networking elements that posses both the control and the data plane in every element that is installed in the network. This puts a slight disadvantage as setting up a new element would require making changes in control plane of almost every element that exists in the network. This leads to decrease in optimality goal of the network.

On the other hand the control plane from every element of the network is detached and the SDN controller is created and the elements left without the control pane are now termed as OvSwitches that are just used to regulate the flow data packets. This provides more configuration flexibility and accuracy accompanied with Data Flow optimization.

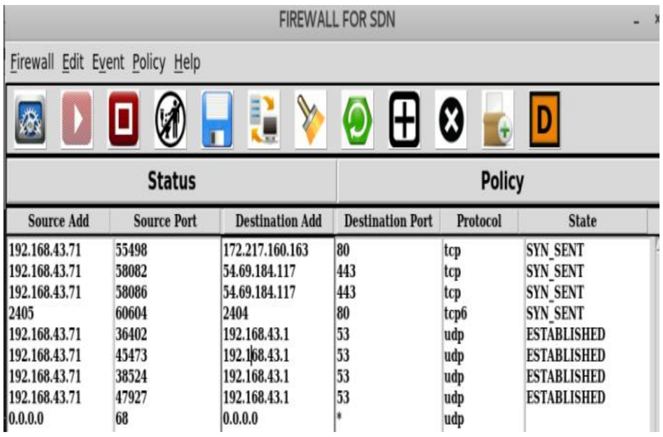
Same thing happens in the case of Firewalls in the traditional and SDN networks. The firewalls in the traditional networks have Limited functionalities where as in SDN more privileges are granted to the same firewall thus providing better security.

## III. THE PROPOSED SYSTEM.

The System proposed is a firewall that is based on the concept of SDN technology i.e. it will perform the functionalities of a firewall based on the fact that it has to safeguard the Controller as well as the network it is working on. It will not only keep a check on the outside network but also the emulated network in the Mininet. Keeping this view in record, the Firewall will be implemented in following steps
1)Coding the firewall in Python Language.
2)Installing Mininet and Creating the Network Topology.
3)Setting up the Firewall (Firewall setup phase).
4)Checking the functionalities(Operation phase)
5)Attack on the Firewall.

### 3.1 CODING THE FIREWALL IN PYTHON LANGUAGE.

Since we will be working with mininet, it supports Python codes and hence the firewall to sit upon the Controller will be coded in Python.
We will use tkinter library of Python for creating the GUI of our firewall.
The GUI will have two major blocks one is the Status which will tell the current network connections of the controller and the network emulated and other will be Policy which will display the rules set up by the user. It will also have different toggles and options for insert update and delete of functionalities and help too.
The Screenshot below shows the GUI created for the firewall.



| Source Add | Source Port | Destination Add | Destination Port | Protocol | State |
|---|---|---|---|---|---|
| 192.168.43.71 | 55498 | 172.217.160.163 | 80 | tcp | SYN_SENT |
| 192.168.43.71 | 58082 | 54.69.184.117 | 443 | tcp | SYN_SENT |
| 192.168.43.71 | 58086 | 54.69.184.117 | 443 | tcp | SYN_SENT |
| 2405 | 60604 | 2404 | 80 | tcp6 | SYN_SENT |
| 192.168.43.71 | 36402 | 192.168.43.1 | 53 | udp | ESTABLISHED |
| 192.168.43.71 | 45473 | 192.168.43.1 | 53 | udp | ESTABLISHED |
| 192.168.43.71 | 38524 | 192.168.43.1 | 53 | udp | ESTABLISHED |
| 192.168.43.71 | 47927 | 192.168.43.1 | 53 | udp | ESTABLISHED |
| 0.0.0.0 | 68 | 0.0.0.0 | * | udp | |

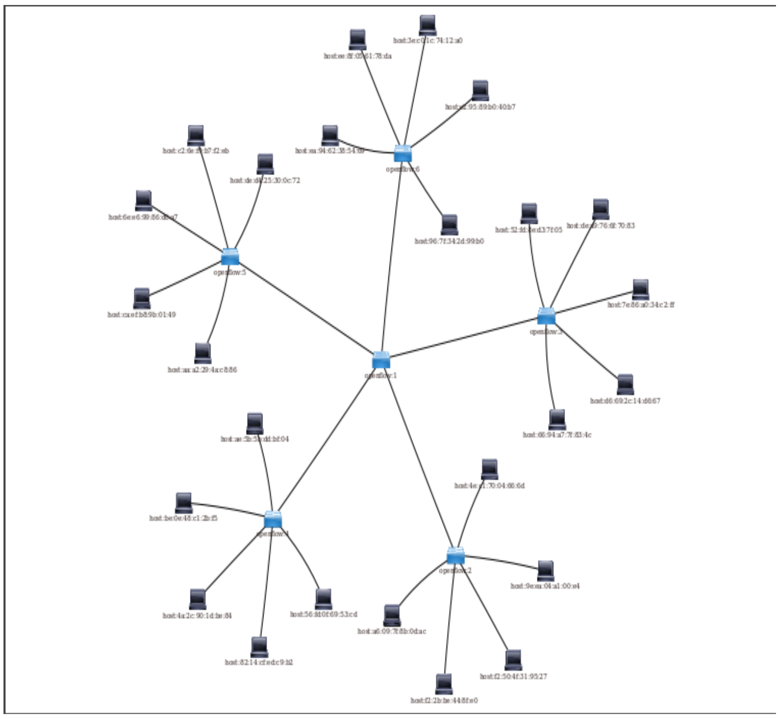
### 3.2 Installing Mininet and Creating the Network Topology

Our project will be emulated in Mininet which is a network emulator which creates a network of virtual hosts, switches, controllers, and links in other sense a fully functioning network as per convenience. The hosts thus created run standard Linux network software, and its switches support highly flexible custom routing i.e SDN.
The Advantages of Mininet are:

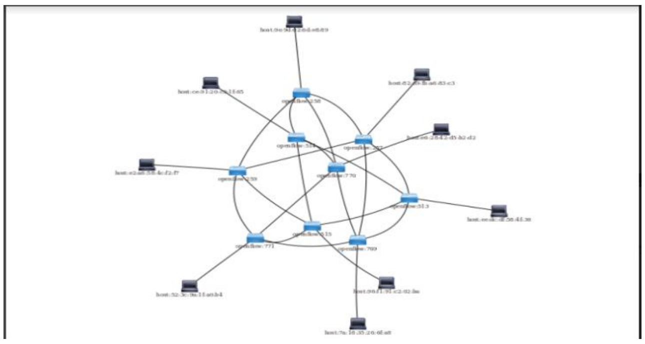
- ➢ It provides a simple and inexpensive network testbed for development and research
- ➢ It allows multiple concurrent developers to work independently in the same network on the same topology.
- ➢ It allows testers to perform complex topology testing, without the need to wire up a physical network and analyze it for proper optimal results.
- ➢ Mininet can be used without programming that is out of the box, but also provides an extensible Python API for network creation and experimentation purposes.
- ➢ It helps in getting an easy way to get correct system behavior(and, to the extent supported by your hardware, performance) and to experiment with topologies. Which can be created or imagined.

These networks which will be created virtually resemble a real Linux kernel and network stack (including any kernel extensions which you may have available, as long as they are compatible with network namespaces.) Thus GUI developed will be tested on Mininet, for an SDN controller, with modified switch, or host,also can be moved to a real system with minimal changes, for real-world testing, performance evaluation, and deployment. Thus indicating that a design that works in Mininet can usually be moved directly to hardware.

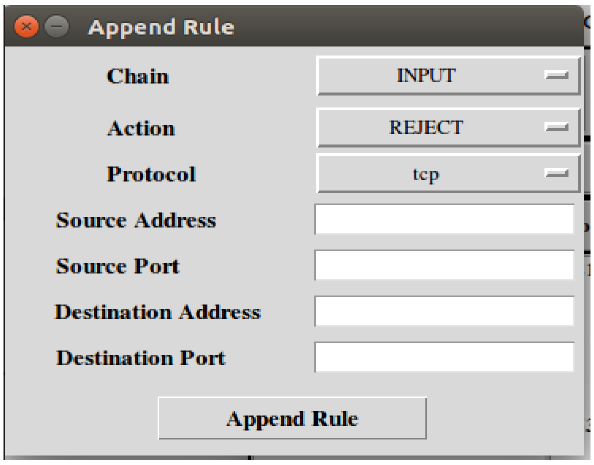
After installation setting up the network is simply work of commands to specify the type of topology and number of nodes. One such Tree topology is show in the picture below.



This Topology has the SDN controller at the center. Each switch including the controller has 5 connections. So in all it has 1 controller 5 switches and 25 nodes.
The command used for the same is
“sudo mn –topo tree,depth=2 fanout=5”
Another example for the same is:



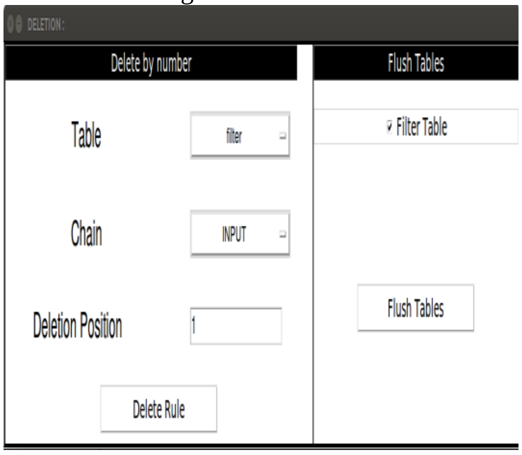
**3.3 Setting the Firewall(Firewall Setup Phase).**
This is obviously the essential step of the project as after the setting of the firewall the rules that guard the firewalls judgment are decided.
In this step the GUI created in step1 is put to actual use. The user uses the GUI to insert proper rules in the iptables of the firewall and and checks the network statistics
Like the traditional networks, here also the iptables consist of Input Output and Forward tables with following functionalities
a)Append:- This adds a new rule to the end of all existing rules.



Append Rule

| | |
|---|---|
| Chain | INPUT |
| Action | REJECT |
| Protocol | tcp |
| Source Address | |
| Source Port | |
| Destination Address | |
| Destination Port | |

b)Delete: this is used to delete a rule.
Where as Flush totally clears all the rules in a single click



DELETION :

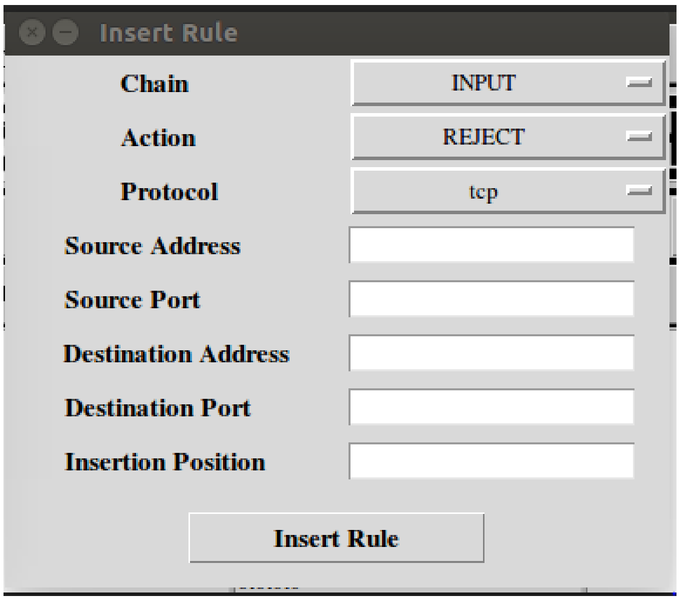
Delete by number

| | |
|---|---|
| Table | filter |
| Chain | INPUT |
| Deletion Position | 1 |

Flush Tables

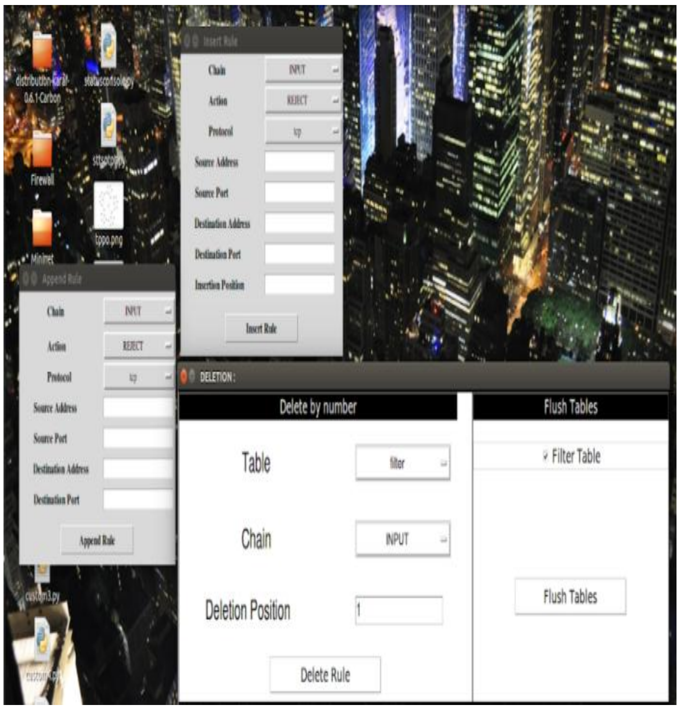
Filter Table

c)Insert: to add a rule at the start .



Insert Rule

| | |
|---|---|
| Chain | INPUT |
| Action | REJECT |
| Protocol | tcp |
| Source Address | |
| Source Port | |
| Destination Address | |
| Destination Port | |
| Insertion Position | |



Insert Rule

| | |
|---|---|
| Chain | INPUT |
| Action | REJECT |
| Protocol | tcp |
| Source Address | |
| Source Port | |
| Destination Address | |
| Destination Port | |
| Insertion Position | |

Append Rule

| | |
|---|---|
| Chain | INPUT |
| Action | REJECT |
| Protocol | tcp |
| Source Address | |
| Source Port | |
| Destination Address | |
| Destination Port | |

DELETION :

Delete by number

| | |
|---|---|
| Table | filter |
| Chain | INPUT |
| Deletion Position | 1 |

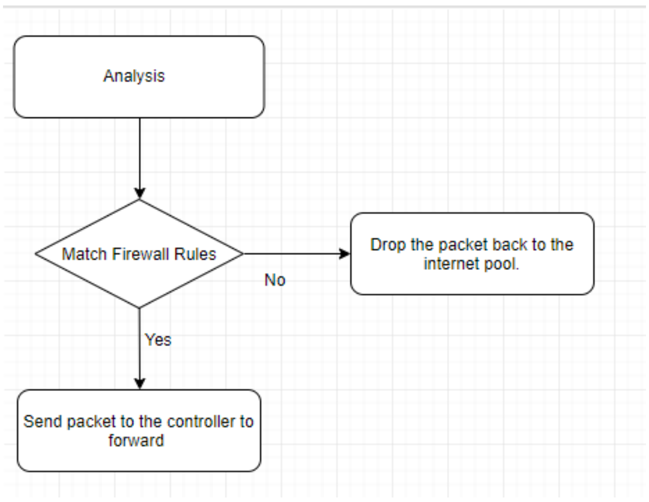
Flush Tables

✓ Filter Table

Like wise it has three functions as
a)Accept: to allow the packets to pass in the network.
b)Drop: to disallow any packet without even any acknowledgment
c)Reject: to disallow any packet with an acknowledgment

### 3.4 Checking the FUNCTIONALITIES (Operational Phase)

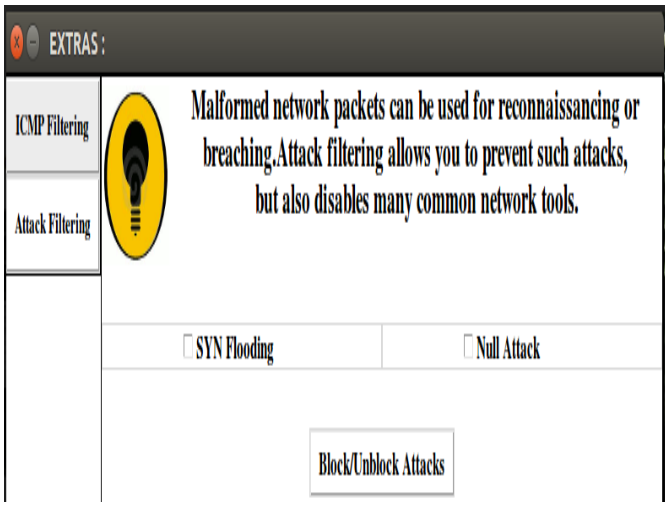
In this step, the firewall that is deployed is put to function. Here in each and every functionality of the Firewall is exploited and checked upon to verify its efficiency. This Step can be depicted diagrammatically as below:


Analysis
Match Firewall Rules
No
Drop the packet back to the internet pool.
Yes
Send packet to the controller to forward

The different steps are:
Step1: Analysis
The packet that arrived at the firewall needs to be analyzed for various factors like the source address, destination address, the network subnet and the headers etc. Each and every aspect that needs to be checked for maintaining integrity of the data packet received and to be assured that the contents are error-less are performed in this very step. The end of this step marks a verified data packet and that packet itself has to face the firewall created by us. The analysis verifies the packet and if non suitable case found, the packet is thrown away into the internet pool and is not sent further for any verification. On the other hand the packet that packet analysis moves ahead to face the wall. Thus analysis proves as the door step verification or as the first line of verification that can be performed by the firewall.
Step 2: Match Firewall Rules:
The second step marks the second line of verification in the project. Here the packet that passed the analysis is put to face the Firewall itself.
The firewall uses the rules that it sets up in the Setup Phase of the project and verifies the packet accordingly. It checks for contents of the packet and check for malicious data, bugs, worms that may or may not be present in the data packet. The rules that are set help the firewall in deciding whether the packet received is suitable for the host or not. The questions like “whether the packet is true or not?” and “whether the packet will cause harm or no harm at all?” are to be answered buy the firewall and then as per the answers it decides the quality of the packet. This quality factor proves helpful in deciding the further fate of the data packet. If the firewall feels that the packet is suitable for the host then it is passed and forwarded to the respective OVSwitch by the SDN controller. On the other hand if it fails to pass, then it is straight away thrown back into the internet pool and left there to travel until its TTL comes to an end. There also may exist a possibility where in the packet keeps on re-arriving at the same firewall and getting rejected a multiple times by the same procedure.

## 3.5 THE ATTACK

This is the final testing step in the project.
Inhere on purpose a malicious attack will be performed on the firewall to check for its functionalities. The attack can be of any type and the firewall must be able to defend against it.
The attacks performed are:-
a)Syn Flooding:- In here a flow of continued SYN request is sent to the target so as to consume up the resources of the target.
b)Null attack: A null session is an anonymous connection to an inter-process communication network service on Windows-based computers.



Malformed network packets can be used for reconnaissancing or breaching.Attack filtering allows you to prevent such attacks, but also disables many common network tools.

## IV. CONCLUSION

Though Firewall has its own limitations, its a healthy and safe procedure for the betterment of networking in various countries round the globe. Not only our security is increased, but its various problems are solved due to it thus resulting in less preaching and disloyalty. It has a bright future and economic scope thus not only it ll generate employment but also it ll be a generator of money as well. Irrespective of the fact that with new technology new threats are released but dealing with that threats and breaking it down to the very core thus ensuring integrity is the scope of this project.

## REFERENCES

[1] Michelle Suh, Sae Hyong Park, Byungjoon Lee, SunheeYang Building Firewall over the Software-Defined-Network Controller SDN Research Section, ETRI (Elec-tronics and Telecommunications Research Institute),Korea
[2] Jake Collings,Jun Liu An OpenFlow-based Prototype of SDN-Oriented Stateful Hardware Firewalls 2014, IEEE22nd International Conference on Network Protocols
[3] Justin Gregory V. Pena and William Emmanuel Yu Development of a Distributed Firewall Using Soft-ware Defined Networking Technology Department of Information Systems and Computer Science Ateneo De Manila University Loyola Heights, Quezon City,Philippines
[4] Thuy Vinh Tran,Heejune Ahn A Network Topology-aware Selectively Distributed Firewall Control in SDN Department of Electrical and Information Engineering Seoul National University of Science and Technology Seoul, Republic of Korea
[5] https://turbonomic.com/blog/ontechnology/sdn-software-defined-networking-primer-and-why-we-need-sdn/
[6] https://www.pressreader.com/india/opensource-for-you/20160610/282157880544330
[7] https://en.wikipedia.org/wiki/SDN
[8] https://www.slideshare.net/lz1dsb/why-sdn
[9] http://opensourceforu.com/2016/07/implementing-a-software-defined-network-sdn-based-firewall/
[10] http://sdnhub.org/resources/useful-mininet-setups/
[11] https://www.techopedia.com/definition/4038/packet-filtering
[12] http://securityworld.worldiswelcome.com/packet-filtering-firewall-an-introduction
[13] https://www.sdxcentral.com/sdn/definitions/sdn-controllers/opendaylight-controller/
[14] **http://www.necoma-project.eu/m/filer** public/18/30/1830b40c-a2e2-4a0b-b83e-af8b2d846e61/imt-zhang-comnet2015.pdf
[15] https://www.ietf.org/proceedings/91/slides/slides-91-i2nsf-0.pdf

Viren Wadhwani "SDN Based Firewall." IOSR Journal of Engineering (IOSRJEN), vol. 08, no. 6, 2018, pp. 24-30.