CloudFirewall: An SDN based Firewall supporting three-different work modes

J COMPONENT PROJECT REPORT - A1 SLOT

Submitted by,

Arkaprava Mahato 19BCT0173
arkaprava.mahato2019@vitstudent.ac.in

Ayush Wunnava 19BCT0181
ayush.wunnava2019@vitstudent.ac.in

Jaiswal Abhijeet Manjeet 19BCT0191
abhijeet.manjeet2019@vitstudent.ac.in

Course Code: BCT3008
Course Title: Software Defined Networks
Under the guidance of
Prof. Ushus Elizabeth Zachariah
B.Tech - Computer Science and Engineering

School of Computer Science and Engineering
Apr, 2022

Table of Contents

1. ABSTRACT

Software Defined Networking has many advantages over traditional networking one of such advantage is SDN has separated the data plane and control plane so everything in data plane will be controlled through control plane however it sometime can lead to a single point of failure, so the developers need to be careful while configuring the SDN controller in this Project our Aim is to build a SDN Based Cloud Firewall. The firewall will act as a SDN controller and will forward the packets or won't forward them based on our written rules. We have used POX controller which uses python. And Mininet to create topology and test the forwarding of the packets.

2. 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 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.

Cloud Firewalls are software-based, cloud deployed network devices, built to stop or mitigate unwanted access to private networks. As a new technology, they are designed for modern business needs, and sit within online application environments.

Our SDN Based Cloud Firewall supports three different work modes: black-list based blocking, white-list based forwarding, and a pass-through mode which forwards all traffic, but still gathers different statistics on it. We have also implemented a simple web-based UI which can be used to manage settings and inspect statistics on the network traffic and the firewall's functionality.

Cloud Firewall is implemented as an SDN controller, which is programmed to forward or block certain TCP/UDP flows, where a TCP/UDP flow can be uniquely identified by the five-tuple of < *source IP, destination IP, transport protocol type, source port, destination port* >. This SDN controller controls, using the OpenFlow protocol, an underlying SDN switch which interconnects two different networks.

3. RESEARCH

3.1 LITERATURE REVIEW

[1] Cuppens, N., Zerkane, S., Li, Y., Espes, D., Le Parc, P., Cuppens, F. (2017). Firewall Policies Provisioning Through SDN in the Cloud. In: Livraga, G., Zhu, S. (eds) Data and Applications Security and Privacy XXXI. DBSec 2017. Lecture Notes in Computer Science(), vol 10359. Springer, Cham. https://doi.org/10.1007/978-3-319-61176-1_16

The evolution of the digital world drives cloud computing to be a key infrastructure for data and services. This breakthrough is transforming Software Defined Networking into the cloud infrastructure backbone because of its advantages such as programmability, abstraction and flexibility. As a result, many cloud providers select SDN as a cloud network service and offer it to their customers.
However, due to the rising number of network cloud providers and their security offers, network cloud customers strive to find the best provider candidate who satisfies their security requirements.

SO in this paper The authors have proposed a negotiation and an enforcement framework for SDN firewall policies provisioning. Their solution enables customers and SDN providers to express their firewall policies and to negotiate them via an orchestrator. Then, it reinforces these security requirements using the holistic view of the SDN controllers and it deploys the generated firewall rules into the network elements.
The authors have evaluated the performance of the solution and demonstrated its advantages through this paper

[2]
A. Mahesh, A. Chandrasekaran, R. ArunKumar, K. SivaKumar and N. Vigneshwaran, "Cloud based firewall on OpenFlow SDN network," 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), 2017, pp. 1-6, doi: 10.1109/ICAMMAET.2017.8186699.

Security is a major problem in networking. Corporates and Service providers spend much money on highly priced firewalls to implement security. Software-Defined Networking (SDN) is a new structure that can save Corporates and service providers money, reduce provisioning time from days to minutes, provides centralized management, allow programmability in the implicit Network. SDN accomplishes this by decoupling the control plane from the data plane.

In t his paper, the authors have demonstrated the benefits of implementing a Cloud based firewall on OpenFlow SDN network, which is a way of implementing SDN. The firewall has the capability of identifying application layer traffic, like social Medias and other websites and excluding some Denial-of-Service Attacks.
This paper handles the implementation information and give a performance analysis of the firewall component.

[3] ***Rezaei, Ghazal & Hashemi, Massoud. (2021). An SDN-based Firewall for Networks with Varying Security Requirements. 1-7. 10.1109/CSICC52343.2021.9420571.***

With the new coronavirus crisis, medical devices' workload has increased dramatically, leaving them growingly vulnerable to security threats and in need of a comprehensive solution. In this work, we take advantage of the flexible and highly manageable nature of Software Defined Networks (SDN) to design a thoroughgoing security framework that covers a health organization's various security requirements.

In this paper the authors have given a solution that comes to be an advanced SDN firewall that solves the issues facing traditional firewalls. It enables the partitioning of the organization's network and the enforcement of different filtering and monitoring behaviors on each partition depending on security conditions.

The authors have pursued the network's efficient and dynamic security management with the least human intervention in designing our model which makes it generally qualified to use in networks with different security requirements.

[4] SDN-based Stateful Distributed Firewall Ankur Chowdhary, Dijiang Huang, Adel Alshamrani and Abdulhakim Sabur Arizona State University

Software Defined Networking (SDN) simplifies networking management by decoupling control plane and data plane. The SDN controller can dynamically configure multiple physical or virtual network switches. The lack of built-in security in theSDN limits its adoption, as reported some campus adopters.: There are two main issues we identified in current SDN-based firewall architecture. 1) Most of the existing firewall architectures such as Flowguard [5], FortNOX are centralised in nature. In a large cloud network, SDN controller performance can become extremely slow in the centralised architecture. 2) There is no support for packet state maintenance and multi-tenancy in most of these works, as noted by Dixit et al [9].

In this paper, we address the issue of security issues associated with lateral movement of attacker along the east-west plane in a data center, and packet flooding based data plane attacks. One limitation of this work is that we utilise SDFW to showcase defense against layer 4 security attacks. However, a next generation firewall can also act as an application firewall and Deep-Packet-Inspection (DPI) module**.**

[5] Challenges and Preparedness of SDN-based FirewallsConference: the 2018 ACM International Workshop Vaibhav Hemant Dixit, Sukwha Kyung,Ziming Zhao, Adam Doupé, Arizona State University, Tempe, AZ, USA

According to our study, existing defence mechanisms, particularly SDN-based firewalls, face new and SDN-specific challenges in successfully enforcing security policies in the underlying network. In this paper, we identify problems associated with SDN-based firewalls, such as ambiguous flow path calculations and poor scalability in large networks. We survey existing SDN-based firewall designs and their shortcomings in protecting a dynamically scaling network like a data center. We extend our study by evaluating one such SDNspecific security solution called FlowGuard, and identifying new attack vectors and vulnerabilities. We also present corresponding threat detection techniques and respective mitigation strategies**.**
In this work we have juxtaposed existing SDN-based firewalls against each other to inspect their readiness to be deployed in enterprise and large scale networks. We have identified various metrics that an SDN-based firewall solution should address. Seven different firewalls are then compared and evaluated against these metrics.

As a case study, we have deployed FlowGuard on the ScienceDMZ network and discovered underlying vulnerabilities in protecting a large scale, complex network. The challenges are individually discussed and possible mitigation measures are proposed. We want to extend the firewall to incorporate advanced features which protect the SDN network from various attacks targeting these layers. We plan to introduce an agnostic and comprehensive firewall solution of our own with such advanced capabilities to protect an SDN network from intrusions before their occurrence.

[6] IMPLEMENTING SOFTWARE DEFINED NETWORKING (SDN) BASED FIREWALL USING POX CONTROLLER Siddhesh Deshmukh, Amey Gawde, & Nitin Nagori. (2021). International Journal of Innovations in Engineering Research and Technology, 8(09), 168–174.

According to security policy, the firewall is interposed between two networks to buffer traffic between them. By implementing rule-based control on packets, a firewall gives security protection. With either hardware or software, or a fusion of both, firewalls may be implemented. Software-Defined Networking (SDN) is an evolving technology that will drive the networks of the next generation. Network managers are given the freedom to introduce their networks. But at the same time, it brings with its new security problems. We need an effective firewall solution to protect SDN networks. The SDN provides network managers with a simple description of the whole layout of the network. It decouples the control and forwarding mechanisms of a network so that it is possible to handle the physical and logical networks separately.

In this paper,this approach facilitates the programmatic and efficient reallocation of network traffic flows to fulfill increasing needs. SDN makes networks completely managed by software applications and provides the hope of shifting the limits of traditional network infrastructures. For implementation of firewall POX controller is used. POX is an open source OpenFlow/Software Oriented Networking (SDN) Controller built on Python. For quicker design and development of experimental network technologies, POX is used. The POX controller arrives with the Mininet virtual machine pre-installed. Using the POX Controller, we built a Layer-2 Firewall in this project. We were able to manage the switches using POX utilizing rules describing Layer-2 attributes to either permit or prohibit communication between hosts. SDN is a game-changer not just in terms of making control more flexible and controllable, but also in terms of achieving programmability in firewalls by isolating the firewall hardware from the control software.

3.2. SURVEY TABLE

S.No	Advantages	Disadvantages	Method Used	Year
[1]	Novel Negotiation framework is proposed	Firewall rules are open so there is a chance of intrusion	Custom firewall policies And Pox Controller	2017
[2]	Cloud firewall is benefited by openflow protocol	Denial of service attack is not countered by this firewall	Openflow Protocol Cloud firewall	2017
[3]	Human intervention is least	Organizations network partitioning has to be enabled	Cloud firewall, Pox Network slicing	2021
[4]	SDFW scales well on a large network with limited performance impact ~ 1.6% reduction in network bandwidth	A class of DoS attacks can forge the OpenFlow fields with random values, that will lead to table-miss event in the switch.	Openflow Protocol,Local DFW-Event-Listener	2018
[5]	FlowGuard is relatively	Ambiguous Flow	FortNOX,	2018

	more agnostic in nature than other to calculate rule interdependencies	Path Space, Conflicting Priority Handling,Coarse Conflict Resolution	Flowguard,SE-FloodLight	
[6]	Can be used with real hardware, test beds, or the Mininet emulator	firewall does not maintain track of the status of the connection, making it stateless	Openflow Protocol, POX controller	2021

3.3. GAPS IDENTIFIED

Difference Between Traditional Firewall and SDN based Firewall

- internal traffic is not seen and cannot be filtered by a traditional firewall.
- An SDN based firewall works both as a packet filter and a policy checker.
- The first packet goes through the controller and is filtered by the SDN firewall.
- The subsequent packets of the flow directly match the flow policy defined in the controller
- The firewall policy is centrally defined and enforced at the controller.

3.4 PROBLEM STATEMENT

To tackle security issues in SDN by building a SDN based Cloud Firewall

4 PROPOSED METHOD

OVERVIEW OF PROPOSED SYSTEM

Our Aim Is to build a SDN Based firewall. So, in it the firewall will be used as a SDN controller. Project our Aim is to build a SDN Based Cloud Firewall. The firewall will act as a SDN controller and will forward the packets or won’t forward them based on our written rules. We have used POX controller which uses python. And Mininet to create topology and test the forwarding of the packets.

Introduction to Cloud firewall

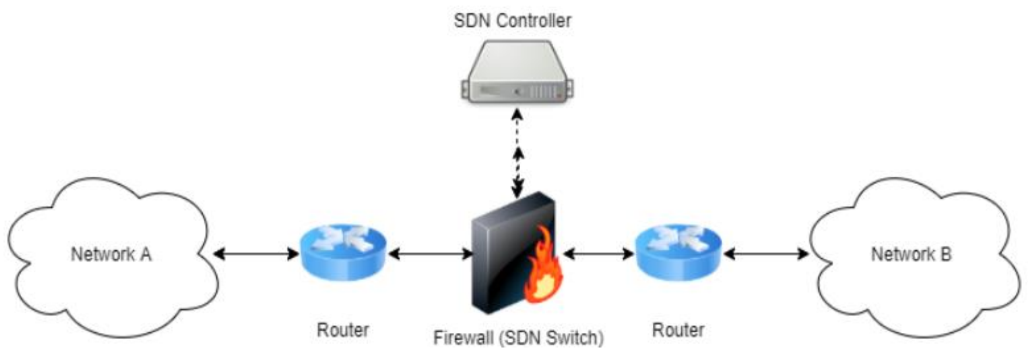
Cloud Firewall is a simple, SDN based firewall, which can be used in order to forward or block certain types of traffic between two different networks. It supports three different work modes: black-list based blocking, white-list based forwarding, and a pass-through mode which forwards all traffic, but still gathers different statistics on it. It also features a simple web-based UI which can be used to manage settings and inspect statistics on the network traffic and the firewall's functionality.

Working

Cloud Firewall is implemented as an SDN controller, which is programmed to forward or block certain TCP/UDP flows, where a TCP/UDP flow can be uniquely identified by the five-tuple of < source IP, destination IP, transport protocol type, source port, destination port >. This SDN controller controls, using the OpenFlow protocol, an underlying SDN switch which interconnects two different networks.

Whenever a packet starting a new flow is received at this switch, it forwards it to the controller, which in turn decides whether this flow should be forwarded to the other network or otherwise blocked altogether. This decision based upon the firewall's current work mode (white-list / black-list / pass-through) and its current defined rules set. When such a decision is made by the controller, it installs an appropriate forwarding rule in the switch so that future packets belonging to the same flow will be handled in the same manner.

Architecture Model



5 SOFTWARES USED

We have Implemented The project on Ubuntu 20.0

For Cloud firewall

- o POX
- o Mininet Wi-Fi
- o OpenFlow
- o Python

For UI

- o HTML
- o JavaScript
- o CSS

6 IMPLEMENTATION

The implementation consists of two different parts:

The SDN firewall:

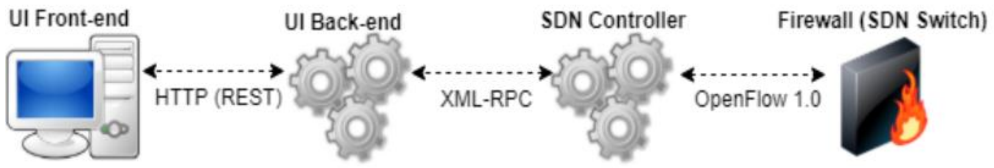
As explained above, the firewall is implemented as an SDN controller. It is written in Python and is built above the POX framework. It exposes an XML-RPC based API which allows manipulating the firewall's behavior (i.e.: changing the firewall's work mode, adding and removing forwarding rules)**.**

The web UI:

The firewall's UI is implemented as a web application. It's back-end is written in Python above the Flask microframework. It exposes a RESTful API which allows manipulating the firewall's settings, i.e.: changing its current work mode, adding or removing forwarding rules, etc. It also allows querying for certain statistical and event-based information regarding the traffic passed through the firewall (i.e.:

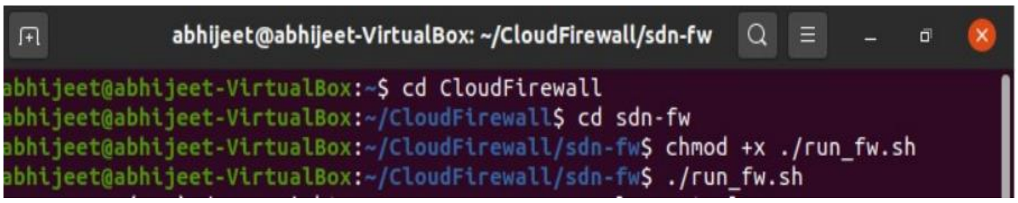
detailed information on flows that were recently blocked by the firewall). One can experiment with the RESTful API by invoking the api_tester.py script.

The front-end is implemented as a single page application and is written in HTML/CSS/JS.



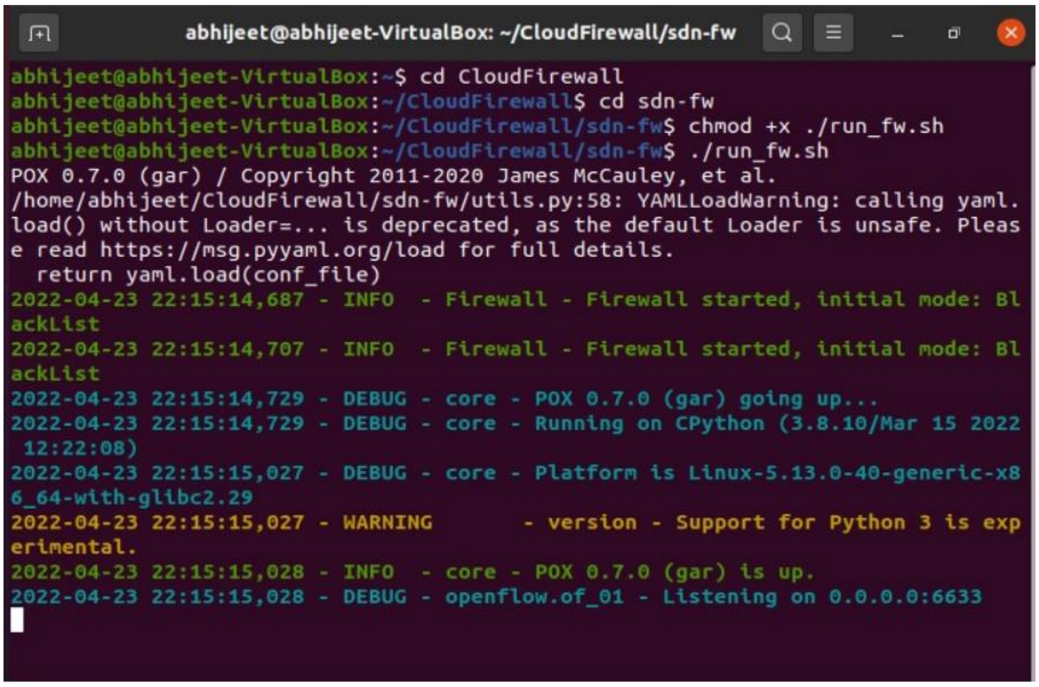
7. RESULTS

Starting The Firewall



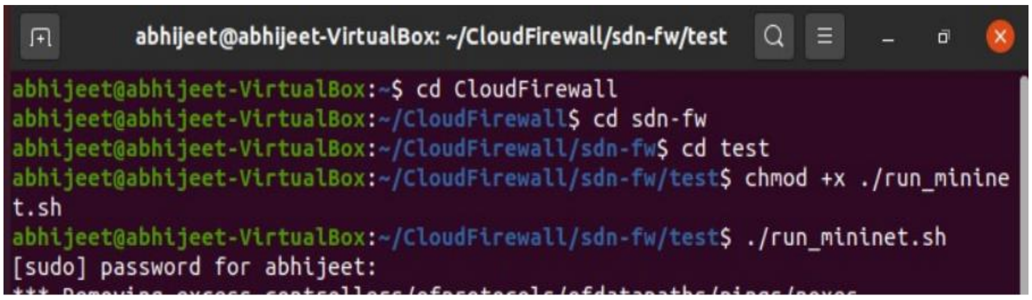
```
abhijeet@abhijeet-VirtualBox:~$ cd CloudFirewall
abhijeet@abhijeet-VirtualBox:~/CloudFirewall$ cd sdn-fw
abhijeet@abhijeet-VirtualBox:~/CloudFirewall/sdn-fw$ chmod +x ./run_fw.sh
abhijeet@abhijeet-VirtualBox:~/CloudFirewall/sdn-fw$ ./run_fw.sh
```

Started The Firewall



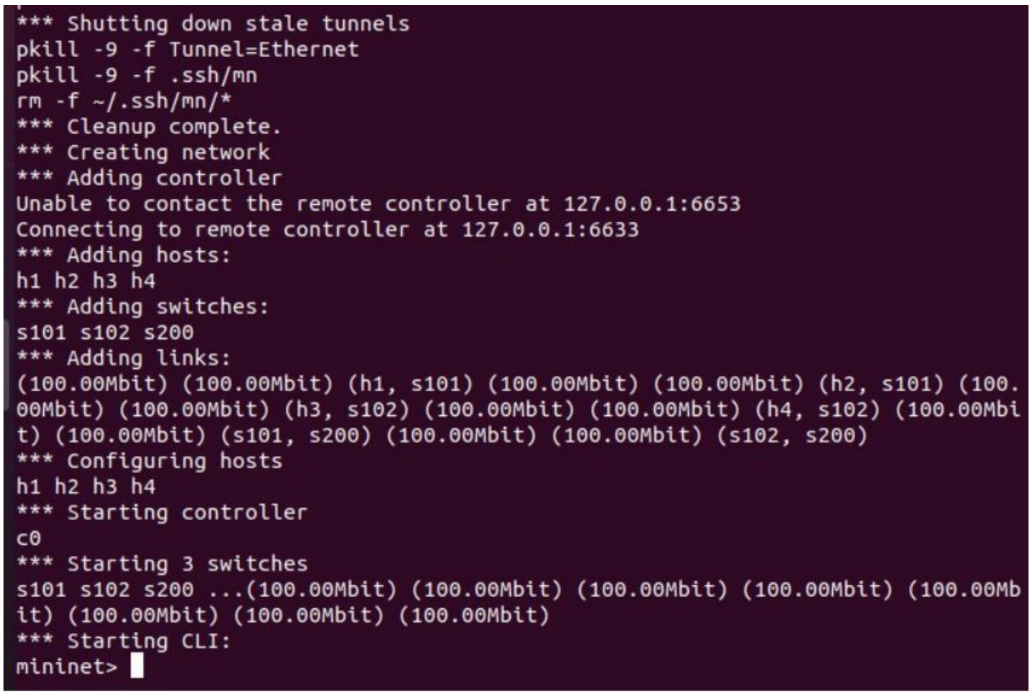
```
abhijeet@abhijeet-VirtualBox:~$ cd CloudFirewall
abhijeet@abhijeet-VirtualBox:~/CloudFirewall$ cd sdn-fw
abhijeet@abhijeet-VirtualBox:~/CloudFirewall/sdn-fw$ chmod +x ./run_fw.sh
abhijeet@abhijeet-VirtualBox:~/CloudFirewall/sdn-fw$ ./run_fw.sh
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
/home/abhijeet/CloudFirewall/sdn-fw/utils.py:58: YAMLLoadWarning: calling yaml.
load() without Loader=... is deprecated, as the default Loader is unsafe. Pleas
e read https://msg.pyyaml.org/load for full details.
  return yaml.load(conf_file)
2022-04-23 22:15:14,687 - INFO  - Firewall - Firewall started, initial mode: Bl
ackList
2022-04-23 22:15:14,707 - INFO  - Firewall - Firewall started, initial mode: Bl
ackList
2022-04-23 22:15:14,729 - DEBUG - core - POX 0.7.0 (gar) going up...
2022-04-23 22:15:14,729 - DEBUG - core - Running on CPython (3.8.10/Mar 15 2022
 12:22:08)
2022-04-23 22:15:15,027 - DEBUG - core - Platform is Linux-5.13.0-40-generic-x8
6_64-with-glibc2.29
2022-04-23 22:15:15,027 - WARNING       - version - Support for Python 3 is exp
erimental.
2022-04-23 22:15:15,028 - INFO  - core - POX 0.7.0 (gar) is up.
2022-04-23 22:15:15,028 - DEBUG - openflow.of_01 - Listening on 0.0.0.0:6633
```

Starting The Mininet



```
abhijeet@abhijeet-VirtualBox:~$ cd CloudFirewall
abhijeet@abhijeet-VirtualBox:~/CloudFirewall$ cd sdn-fw
abhijeet@abhijeet-VirtualBox:~/CloudFirewall/sdn-fw$ cd test
abhijeet@abhijeet-VirtualBox:~/CloudFirewall/sdn-fw/test$ chmod +x ./run_minine
t.sh
abhijeet@abhijeet-VirtualBox:~/CloudFirewall/sdn-fw/test$ ./run_mininet.sh
[sudo] password for abhijeet:
```

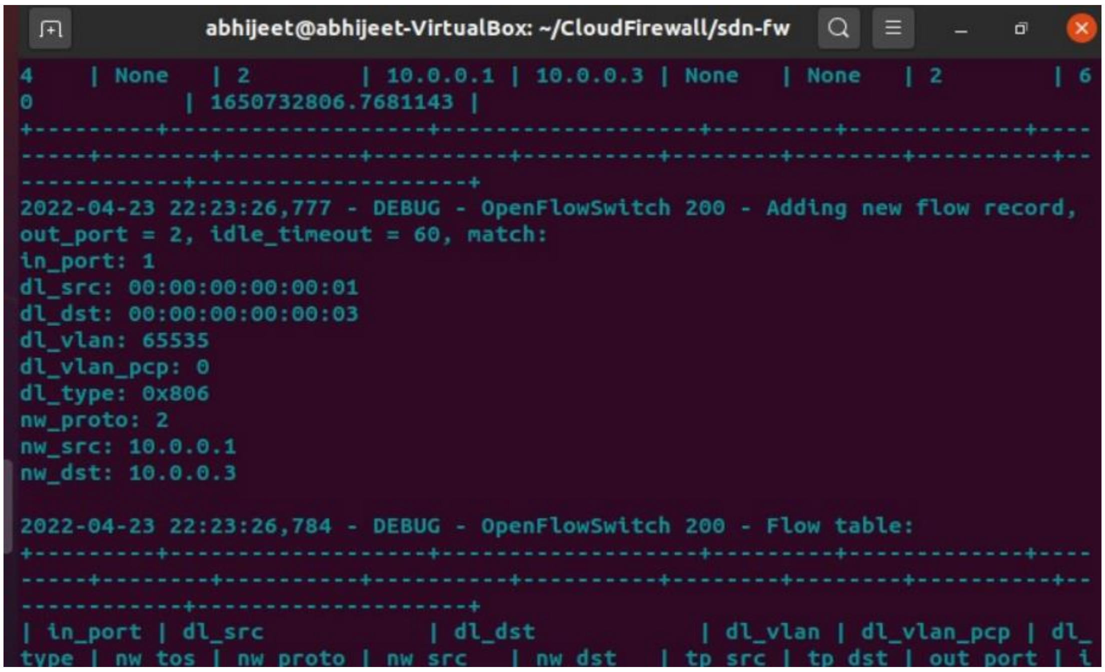
Connecting To Our Mininet Topology



```
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Connecting to remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s101 s102 s200
*** Adding links:
(100.00Mbit) (100.00Mbit) (h1, s101) (100.00Mbit) (100.00Mbit) (h2, s101) (100.
00Mbit) (100.00Mbit) (h3, s102) (100.00Mbit) (100.00Mbit) (h4, s102) (100.00Mbi
t) (100.00Mbit) (s101, s200) (100.00Mbit) (100.00Mbit) (s102, s200)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s101 s102 s200 ...(100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mbit) (100.00Mb
it) (100.00Mbit) (100.00Mbit) (100.00Mbit)
*** Starting CLI:
mininet>
```

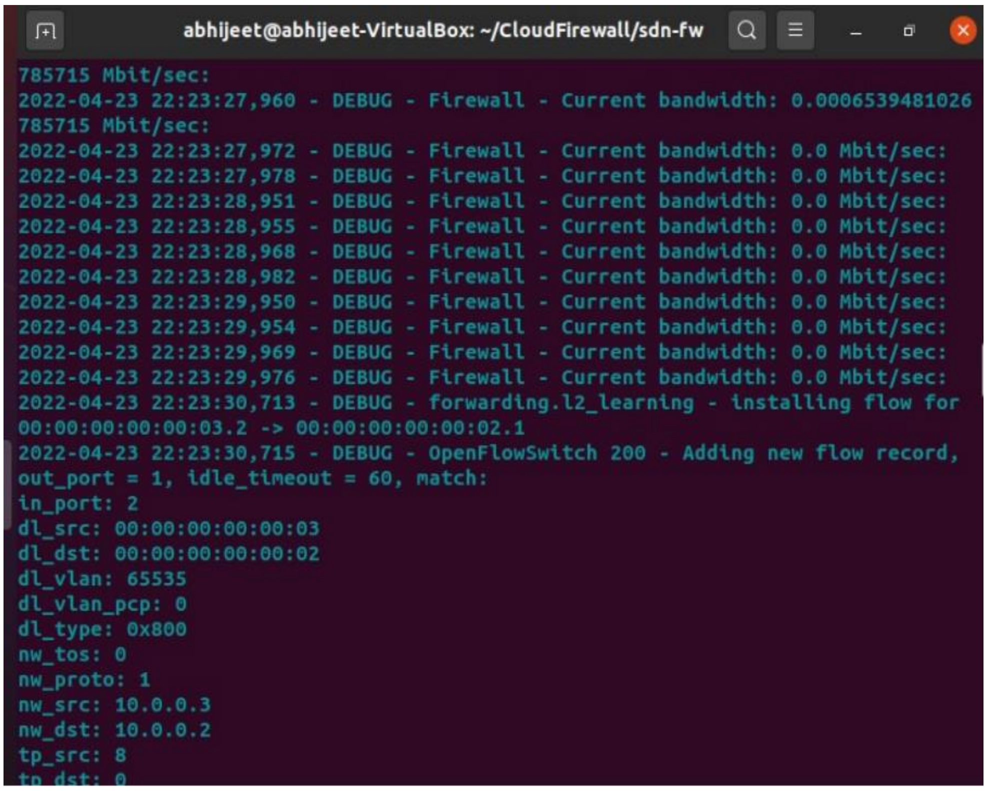
Ping All and The SDN based Firewall will allow or block packet based on written Logic(rules)

Entering the data into Flow table

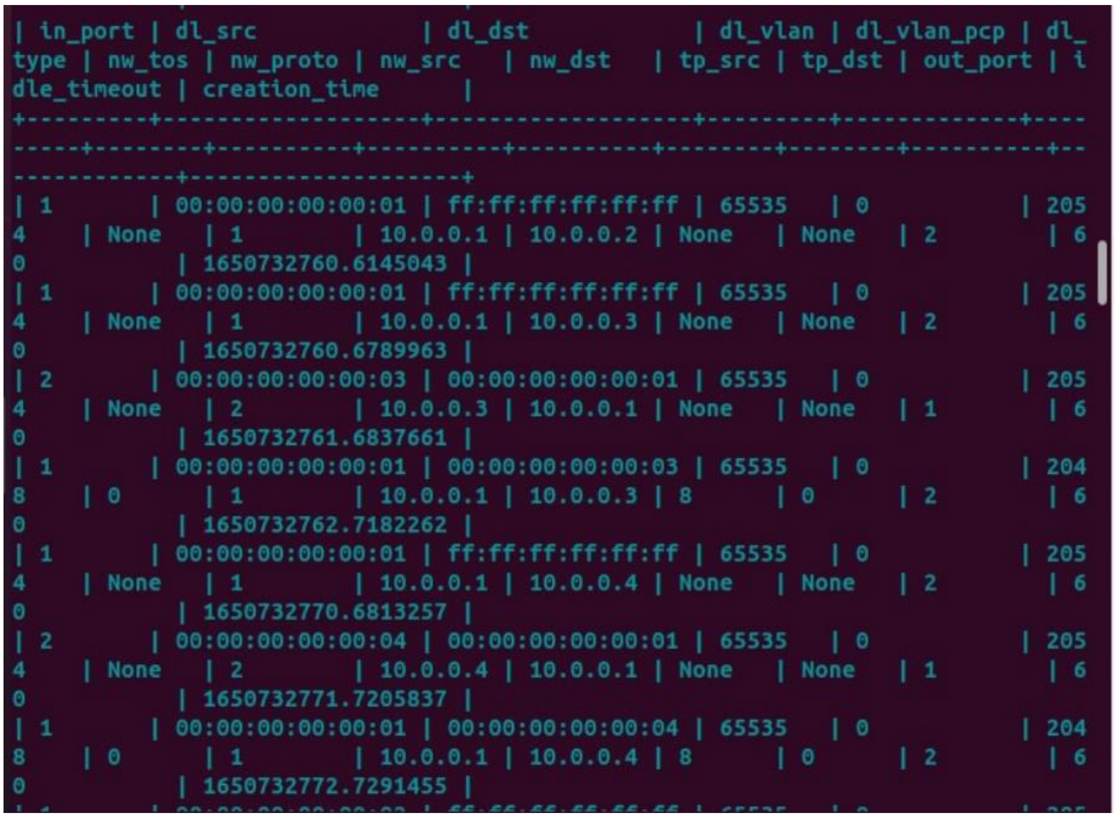


```
4     | None    | 2          | 10.0.0.1 | 10.0.0.3 | None    | None    | 2          | 6
0           | 1650732806.7681143 |
+---------+-------------------+-------------------+---------+--------------+----
-----+--------+---------+---------+---------+--------+--------+----------+--
-----------+-------------------+
2022-04-23 22:23:26,777 - DEBUG - OpenFlowSwitch 200 - Adding new flow record,
out_port = 2, idle_timeout = 60, match:
in_port: 1
dl_src: 00:00:00:00:00:01
dl_dst: 00:00:00:00:00:03
dl_vlan: 65535
dl_vlan_pcp: 0
dl_type: 0x806
nw_proto: 2
nw_src: 10.0.0.1
nw_dst: 10.0.0.3

2022-04-23 22:23:26,784 - DEBUG - OpenFlowSwitch 200 - Flow table:
+---------+-------------------+-------------------+---------+--------------+----
-----+--------+---------+---------+---------+--------+--------+----------+--
-----------+-------------------+
| in_port | dl_src            | dl_dst            | dl_vlan | dl_vlan_pcp | dl_
type | nw_tos | nw_proto | nw_src   | nw_dst   | tp_src | tp_dst | out_port | i
```

The data is being entered into flow table



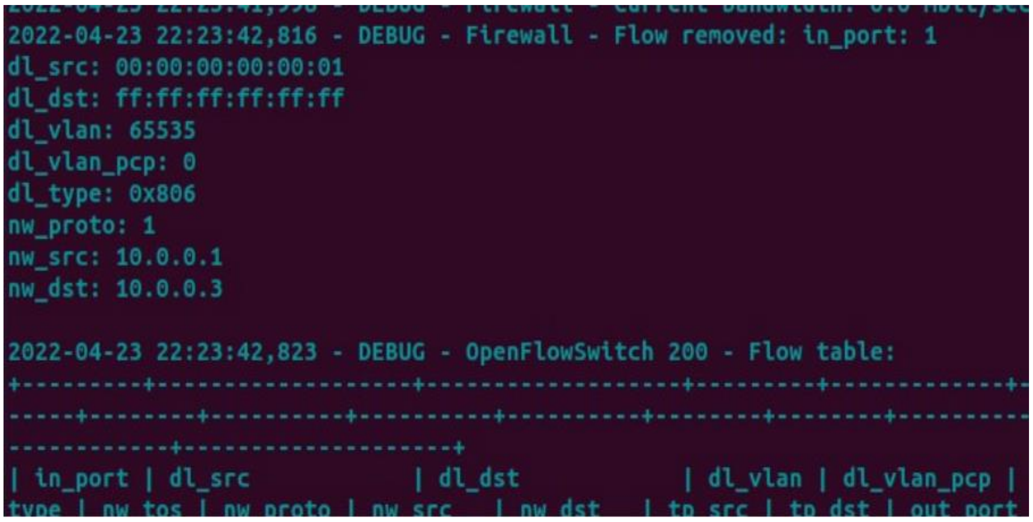
```
785715 Mbit/sec:
2022-04-23 22:23:27,960 - DEBUG - Firewall - Current bandwidth: 0.0006539481026
785715 Mbit/sec:
2022-04-23 22:23:27,972 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:27,978 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:28,951 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:28,955 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:28,968 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:28,982 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:29,950 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:29,954 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:29,969 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:29,976 - DEBUG - Firewall - Current bandwidth: 0.0 Mbit/sec:
2022-04-23 22:23:30,713 - DEBUG - forwarding.l2_learning - installing flow for
00:00:00:00:00:03.2 -> 00:00:00:00:00:02.1
2022-04-23 22:23:30,715 - DEBUG - OpenFlowSwitch 200 - Adding new flow record,
out_port = 1, idle_timeout = 60, match:
in_port: 2
dl_src: 00:00:00:00:00:03
dl_dst: 00:00:00:00:00:02
dl_vlan: 65535
dl_vlan_pcp: 0
dl_type: 0x800
nw_tos: 0
nw_proto: 1
nw_src: 10.0.0.3
nw_dst: 10.0.0.2
tp_src: 8
tp_dst: 0
```



```
| in_port | dl_src            | dl_dst            | dl_vlan | dl_vlan_pcp | dl_
type | nw_tos | nw_proto | nw_src   | nw_dst   | tp_src | tp_dst | out_port | i
dle_timeout | creation_time      |
+---------+-------------------+-------------------+---------+-------------+----
-----+--------+----------+----------+----------+--------+--------+----------+--
-----------+--------------------+
| 1       | 00:00:00:00:00:01 | ff:ff:ff:ff:ff:ff | 65535   | 0           | 205
4    | None   | 1        | 10.0.0.1 | 10.0.0.2 | None   | None   | 2        | 6
0           | 1650732760.6145043 |
| 1       | 00:00:00:00:00:01 | ff:ff:ff:ff:ff:ff | 65535   | 0           | 205
4    | None   | 1        | 10.0.0.1 | 10.0.0.3 | None   | None   | 2        | 6
0           | 1650732760.6789963 |
| 2       | 00:00:00:00:00:03 | 00:00:00:00:00:01 | 65535   | 0           | 205
4    | None   | 2        | 10.0.0.3 | 10.0.0.1 | None   | None   | 1        | 6
0           | 1650732761.6837661 |
| 1       | 00:00:00:00:00:01 | 00:00:00:00:00:03 | 65535   | 0           | 204
8    | 0      | 1        | 10.0.0.1 | 10.0.0.3 | 8      | 0      | 2        | 6
0           | 1650732762.7182262 |
| 1       | 00:00:00:00:00:01 | ff:ff:ff:ff:ff:ff | 65535   | 0           | 205
4    | None   | 1        | 10.0.0.1 | 10.0.0.4 | None   | None   | 2        | 6
0           | 1650732770.6813257 |
| 2       | 00:00:00:00:00:04 | 00:00:00:00:00:01 | 65535   | 0           | 205
4    | None   | 2        | 10.0.0.4 | 10.0.0.1 | None   | None   | 1        | 6
0           | 1650732771.7205837 |
| 1       | 00:00:00:00:00:01 | 00:00:00:00:00:04 | 65535   | 0           | 204
8    | 0      | 1        | 10.0.0.1 | 10.0.0.4 | 8      | 0      | 2        | 6
0           | 1650732772.7291455 |
```

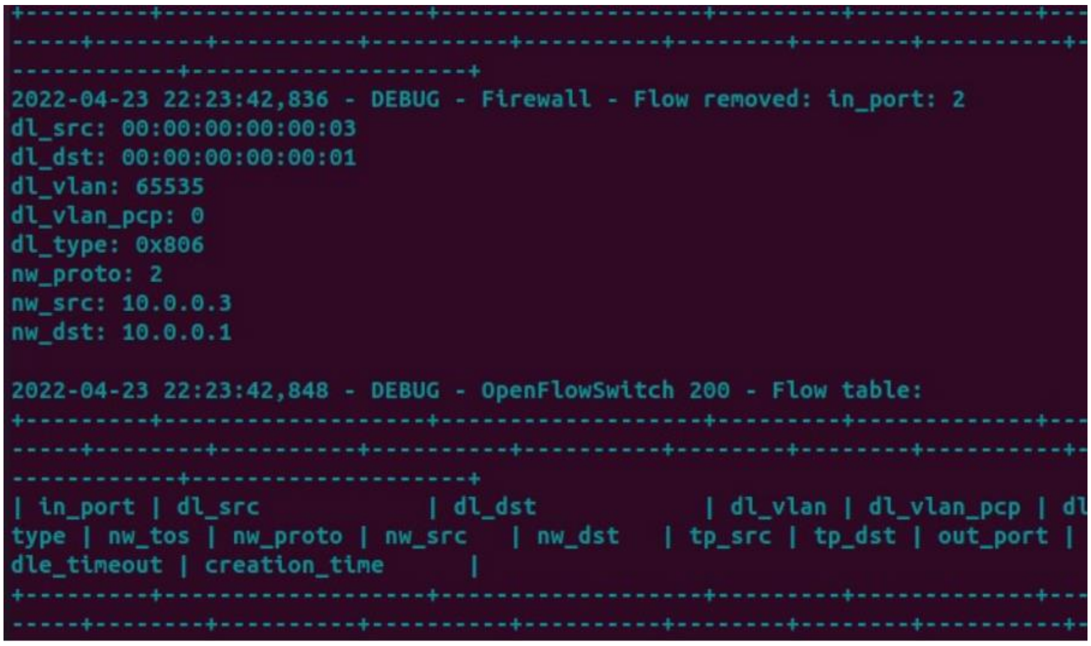
Now The SDN Firewall will not allow the blacklisted IP address to pass through

Removing The blacklisted Entry (Not allowing)

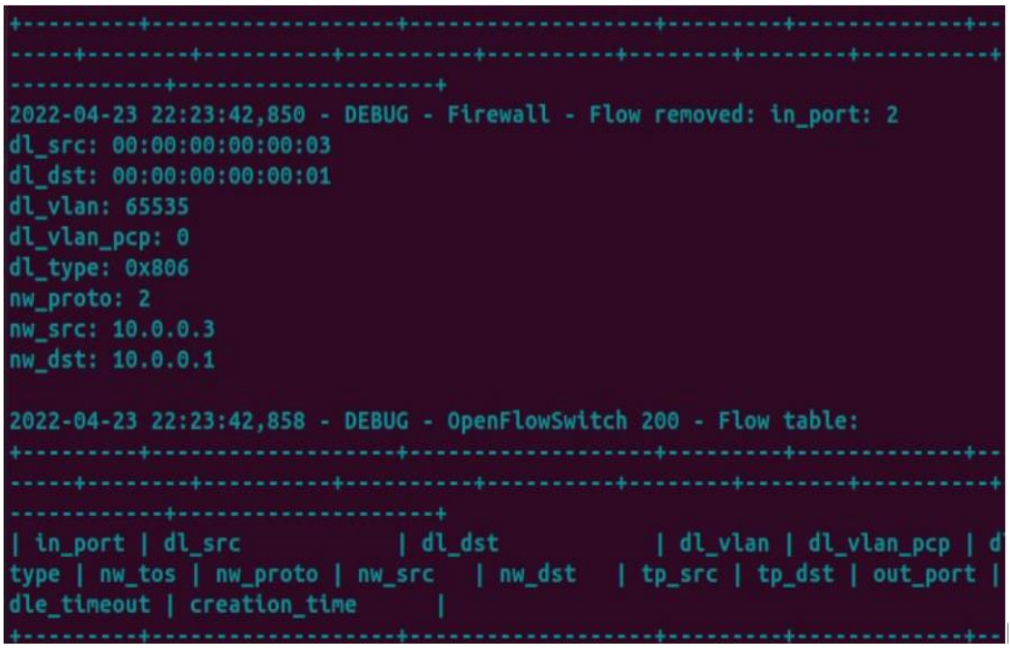


```
2022-04-23 22:23:42,816 - DEBUG - Firewall - Flow removed: in_port: 1
dl_src: 00:00:00:00:00:01
dl_dst: ff:ff:ff:ff:ff:ff
dl_vlan: 65535
dl_vlan_pcp: 0
dl_type: 0x806
nw_proto: 1
nw_src: 10.0.0.1
nw_dst: 10.0.0.3

2022-04-23 22:23:42,823 - DEBUG - OpenFlowSwitch 200 - Flow table:
+---------+-------------------+-------------------+---------+-------------+-
-----+--------+---------+---------+----------+--------+--------+---------
------------+--------------------+
| in_port | dl_src            | dl_dst            | dl_vlan | dl_vlan_pcp |
type | nw_tos | nw_proto | nw_src   | nw_dst   | tp_src | tp_dst | out_port
```

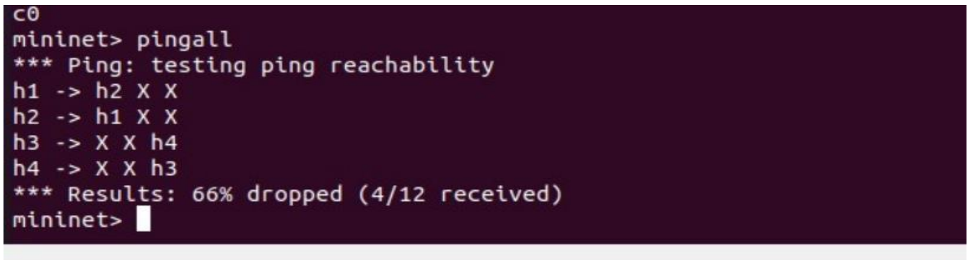
Removing blacklisted Entry(Not allowing)



```
+---------+-------------------+-------------------+---------+-------------+---
-----+--------+---------+---------+----------+--------+--------+----------+-
------------+--------------------+
2022-04-23 22:23:42,836 - DEBUG - Firewall - Flow removed: in_port: 2
dl_src: 00:00:00:00:00:03
dl_dst: 00:00:00:00:00:01
dl_vlan: 65535
dl_vlan_pcp: 0
dl_type: 0x806
nw_proto: 2
nw_src: 10.0.0.3
nw_dst: 10.0.0.1

2022-04-23 22:23:42,848 - DEBUG - OpenFlowSwitch 200 - Flow table:
+---------+-------------------+-------------------+---------+-------------+---
-----+--------+---------+---------+----------+--------+--------+----------+-
------------+--------------------+
| in_port | dl_src            | dl_dst            | dl_vlan | dl_vlan_pcp | dl
type | nw_tos | nw_proto | nw_src   | nw_dst   | tp_src | tp_dst | out_port |
dle_timeout | creation_time      |
+---------+-------------------+-------------------+---------+-------------+---
-----+--------+---------+---------+----------+--------+--------+----------+-
```



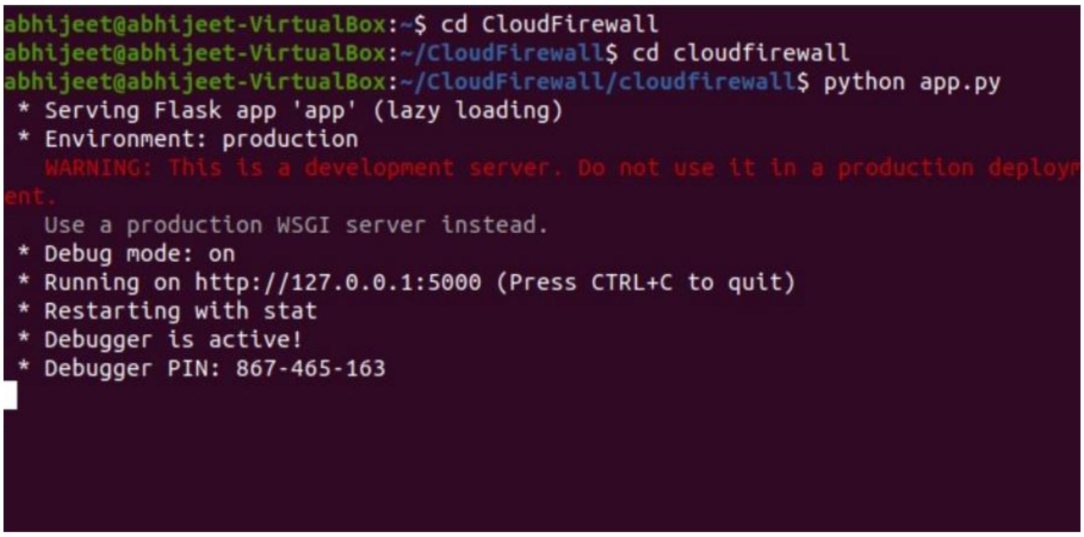
```
+---------+------------------+------------------+---------+-------------+--
-----+--------+----------+----------+----------+--------+--------+---------+
-----------+-------------------+
2022-04-23 22:23:42,850 - DEBUG - Firewall - Flow removed: in_port: 2
dl_src: 00:00:00:00:00:03
dl_dst: 00:00:00:00:00:01
dl_vlan: 65535
dl_vlan_pcp: 0
dl_type: 0x806
nw_proto: 2
nw_src: 10.0.0.3
nw_dst: 10.0.0.1

2022-04-23 22:23:42,858 - DEBUG - OpenFlowSwitch 200 - Flow table:
+---------+------------------+------------------+---------+-------------+--
-----+--------+----------+----------+----------+--------+--------+---------+
-----------+-------------------+
| in_port | dl_src           | dl_dst           | dl_vlan | dl_vlan_pcp | d
type | nw_tos | nw_proto | nw_src   | nw_dst   | tp_src | tp_dst | out_port |
dle_timeout | creation_time     |
+---------+------------------+------------------+---------+-------------+--
```



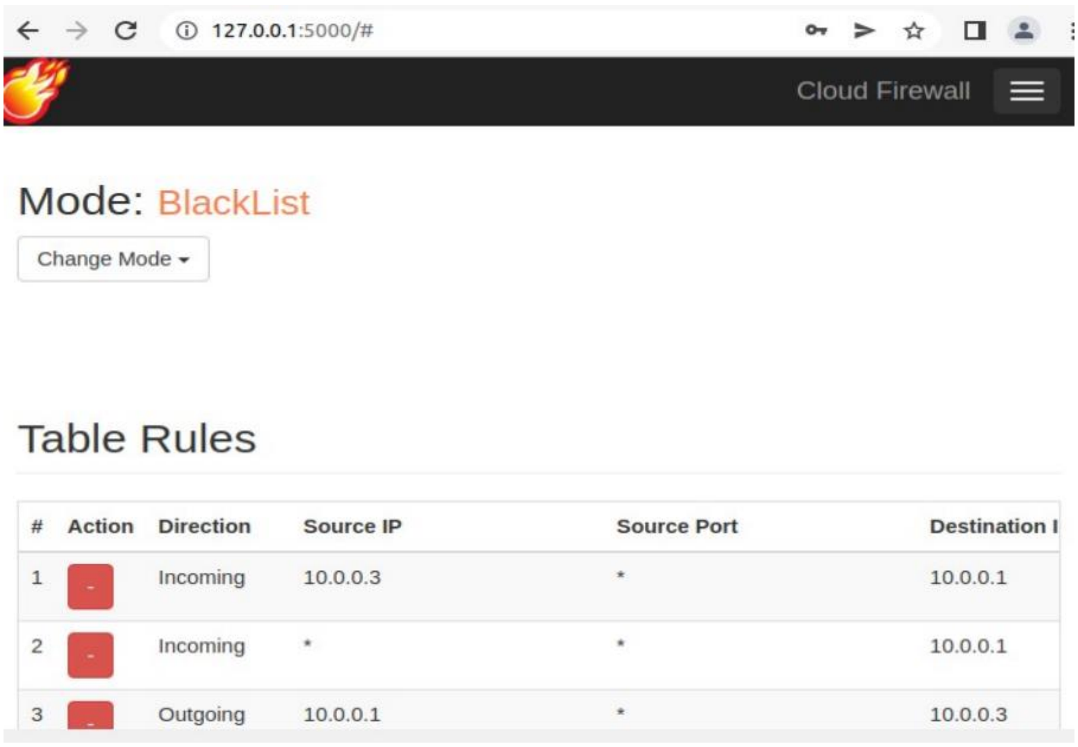
```
c0
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X X
h2 -> h1 X X
h3 -> X X h4
h4 -> X X h3
*** Results: 66% dropped (4/12 received)
mininet>
```

%of Packets dropped (not allowed) by the firewall

Starting the web server



```
abhijeet@abhijeet-VirtualBox:~$ cd CloudFirewall
abhijeet@abhijeet-VirtualBox:~/CloudFirewall$ cd cloudfirewall
abhijeet@abhijeet-VirtualBox:~/CloudFirewall/cloudfirewall$ python app.py
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deploym
ent.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 867-465-163
```



127.0.0.1:5000/#

Cloud Firewall

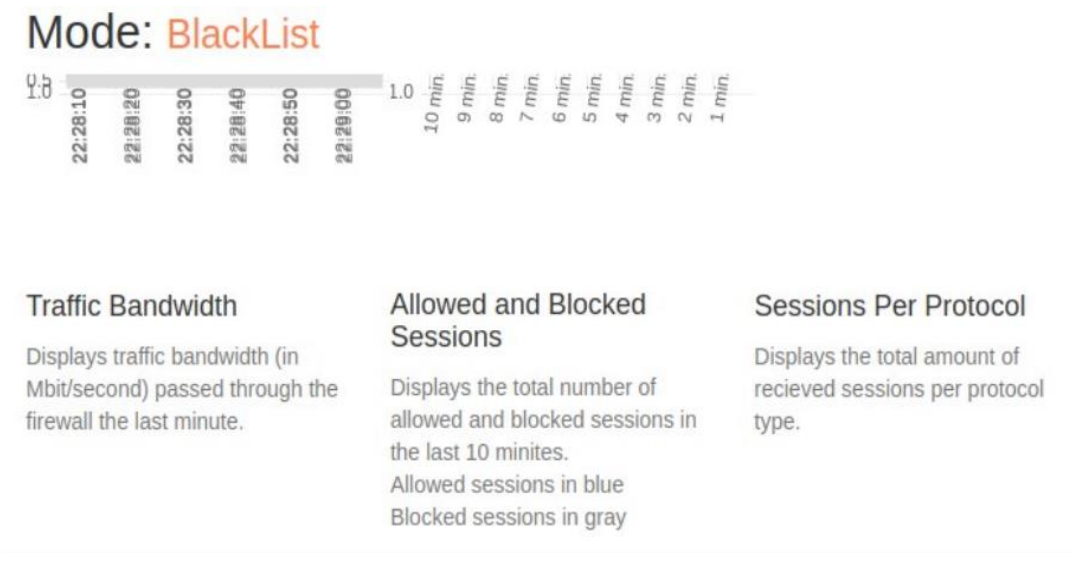
Mode: BlackList

Change Mode

Table Rules

#	Action	Direction	Source IP	Source Port	Destination I
1	-	Incoming	10.0.0.3	*	10.0.0.1
2	-	Incoming	*	*	10.0.0.1
3	-	Outgoing	10.0.0.1	*	10.0.0.3

Visualization



Mode: BlackList

Traffic Bandwidth

Displays traffic bandwidth (in Mbit/second) passed through the firewall the last minute.

Allowed and Blocked Sessions

Displays the total number of allowed and blocked sessions in the last 10 minites.
Allowed sessions in blue
Blocked sessions in gray

Sessions Per Protocol

Displays the total amount of recieved sessions per protocol type.

8.CONCLUSION AND FUTURE WORK

So, we have Implemented A SDN Based Cloud Firewall. We have used Pox framework and Mininet. Also, the code is written in python language the target machine used is Ubuntu. We have successfully implemented three functionalities on our firewall they are whitelist-based forwarding, Blacklist based blocking and a pass-through mode which will allow all the packets to pass.

The Future Scope of the project is we can add multi-cloud functionalities also multi-level security for the firewall

9.REFERENCES

[1] A Comparative Study of traditional Network Firewalls & SDN Firewalls---Gunjan Katwal & Manu Sood,Department of Computer science,Himachal Pradesh University,India

[2] Centralized Firewall for Software-Defined Networking (SDN)--International Research Journal of Engineering and Technology (IRJET), Jishnu Unnikrishnan, Prof. K.S. Charumathi,Pillai College of Engineering, Maharashtra, India

[3] An Enhanced SDN Firewall for Unstructured Local Network--International Journal of Pure and Applied Mathematics,C.R. Greeshma, N.R. Keerthi and Nima S Nair

[4] Challenges and Preparedness of SDN-based Firewalls Vaibhav Hemant Dixit, Sukwha Kyung,Ziming Zhao, Adam Doupé, Yan Shoshitaishvili and Gail-Joon Ahn Arizona State University, Tempe, AZ, USA

[5] J. Li, H. Jiang, W. Jiang, J. Wu and W. Du, "SDN-based Stateful Firewall for Cloud," 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on

Intelligent Data and Security (IDS), 2020, pp. 157-161, doi: 10.1109/BigDataSecurity-HPSC-IDS49724.2020.00037.

[6] IMPLEMENTING SOFTWARE DEFINED NETWORKING (SDN) BASED FIREWALL USING POX CONTROLLER Siddhesh Deshmukh, Amey Gawde, & Nitin Nagori. (2021). International Journal of Innovations in Engineering Research and Technology, 8(09), 168–174.

Appendix-Sample Code

All The code That we have written, is Uploaded on Google Drive

Google Drive Link- SDN Based Firewall Code