SDN based firewall

AIM: To implement firewall on SDN (Software-Defined Networking) network using Python.

INTRODUCTION: A firewall is a system that secures 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 typically maintain a barricade between a confidential, protected internal network and another outside network, such as the Internet, which is assumed not to be secure or trusted. They can be categorised as either hardware or software firewalls. Network firewalls are software programs running on different hardware appliances in the network.

Software based firewalls provide a layer of software on a host, which controls network traffic in and out of that particular machine. Firewall appliances may also provide other functionality to the internal network they protect, such as acting as DHCP or VPN servers for that network. The system analyses data packets for parameters like layer2 or layer3 switch packet formats. It can also perform deep packet scrutiny for higher layer parameters (like application type and services, etc) to filter network traffic. Firewalls are an essential component of any secure network communication for bidirectional packet flow.

Software defined networking (SDN) is the new network technology. It emphasises the separation of the network and the control plane. Responsibility is divided between both the planes. The forward plane is only responsible for packet forwarding in the network. The control plane is responsible for policy creation and its implementation, based on predefined rules. It acts as the entry point of the system and can replace the conventional router. Any packet coming into the network is examined by the control plane and a decision is taken on whether to drop the packet or forward it to the next host. It can also update the IP table entry.

An SDN is the physical separation of the control plane from the data plane. So, instead of each networking device independently forwarding packets to the next hop, the controls are centralised on SDN controllers. SDN is an engaging platform for network virtualization, since each occupant’s control logic can run on a controller rather than on physical switches. It is an approach to computer networking that allows network administrators to manage network services through the abstraction of higher level functionality. So, SDN networks provide flexibility, programmability and simplicity to network operations. Traffic can be directed, adjusted or personalised without requiring physical wiring changes. An SDN promises consolidated control and traffic

management, which deals with automated network security that is more adaptive and mountable.

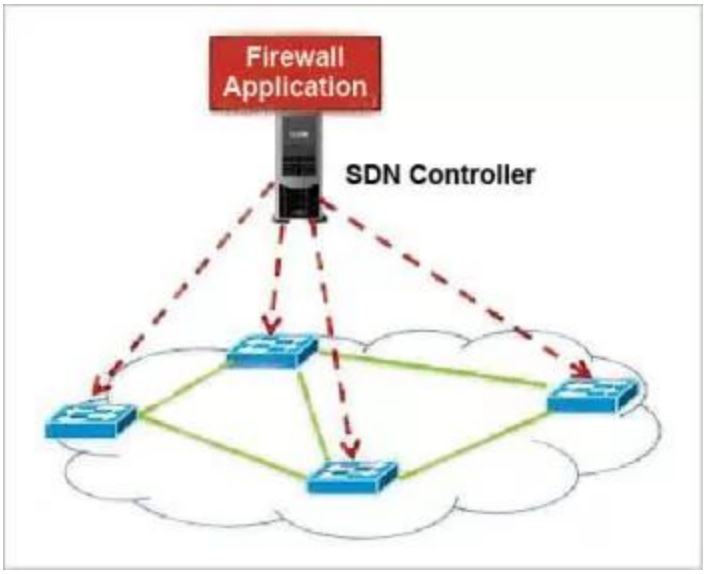


Fig 1: An SDN based firewall

Docker :- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, thanks to the container, the developer can rest assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code.

Docker is a tool that is designed to benefit both developers and system administrators, making it a part of many DevOps (developers + operations) toolchains. For developers, it means that they can focus on writing code without worrying about the system that it will ultimately be running on. It also allows them to get a head start by using one of thousands of programs already designed to run in a Docker container as a part of their application. For operations staff, Docker gives flexibility and potentially reduces the number of systems needed because of its small footprint and lower overhead.

CentOS :- CentOS Linux is a community-supported distribution derived from sources freely provided to the public by Red Hat for Red Hat Enterprise Linux (RHEL). As such, CentOS Linux aims to be functionally compatible with RHEL. The CentOS Project mainly changes packages to remove upstream vendor branding and artwork. CentOS Linux is no-cost and free to redistribute. Each CentOS Linux version is maintained for up to 10 years (by means of security updates -- the duration of the support interval by Red Hat has varied over time with respect to Sources released). A new CentOS Linux version is released approximately every 2 years and each CentOS Linux version is

periodically updated (roughly every 6 months) to support newer hardware. This results in a secure, low-maintenance, reliable, predictable, and reproducible Linux environment.

IMPLEMENTATION DETAILS:

CODE:

GCP :

```
import os     # python "os"  library for executing linux command
command1="iptables -F"      #command1
spoofip=['8.8.8.8']   #dictionary or array
actd="DROP"  #"REJECT"
for i in spoofip:
    print(command1)
    print('iptables -A OUTPUT -d ' +str(i)+ ' -p ICMP --icmp-type 8 -j '+str(actd))
    os.system(command1)
    os.system('iptables -A OUTPUT -d ' +str(i)+ ' -p ICMP --icmp-type 8 -j '+str(actd))
```

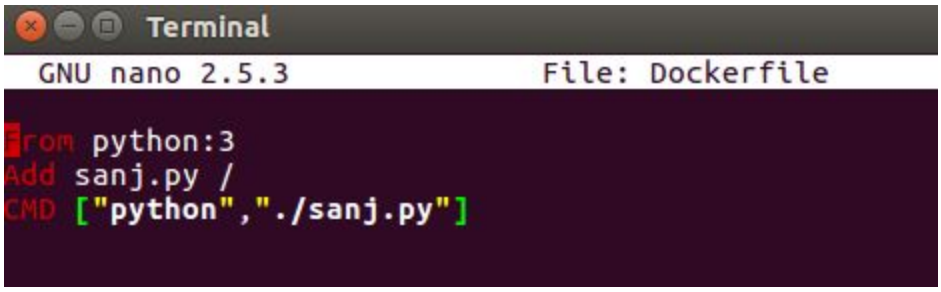
Centos :

```
root@student-OptiPlex-3020:/home/student# docker pull centos
Using default tag: latest
latest: Pulling from library/centos
Digest: sha256:dcbc4e5e7052ea2306eed59563da1fec09196f2ecacbe042acbdcd2b44b05270
Status: Image is up to date for centos:latest
root@student-OptiPlex-3020:/home/student# docker run -ti --rm --cap-add=NET_ADMIN centos
bash
[root@27df14c224be /]# yum install iptables-services
Loaded plugins: fastestmirror, ovl
base                                     | 3.6 kB     00:00
extras                                   | 3.4 kB     00:00
updates                                    | 3.4 kB     00:00
(1/4): base/7/x86_64/group_gz                        | 156 kB   00:05
(2/4): extras/7/x86_64/primary_db                     | 185 kB   00:06
(3/4): base/7/x86_64/primary_db                       | 5.7 MB   00:07
(4/4): updates/7/x86_64/primary_db                      | 6.9 MB   00:07
Determining fastest mirrors
 * base: mirror.dhakacom.com
 * extras: mirror.dhakacom.com
 * updates: mirror.dhakacom.com
Resolving Dependencies
--> Running transaction check
---> Package iptables-services.x86_64 0:1.4.21-18.3.el7_4 will be installed
```

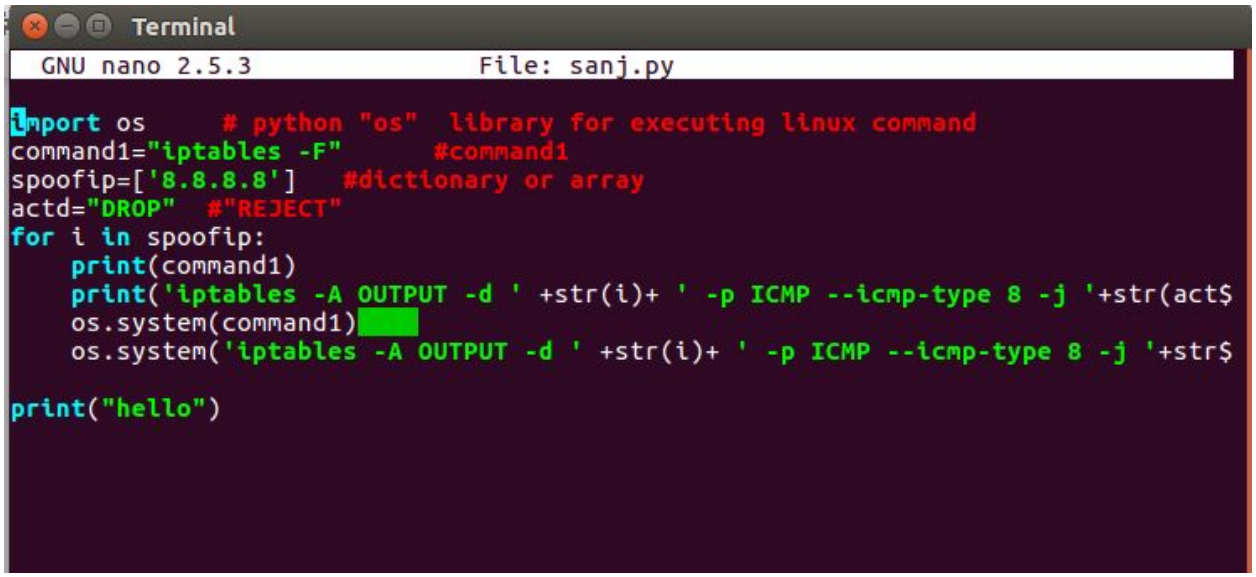
```
--> Processing Dependency: iptables = 1.4.21-18.3.el7_4 for package:
iptables-services-1.4.21-18.3.el7_4.x86_64
--> Running transaction check
---> Package iptables.x86_64 0:1.4.21-18.3.el7_4 will be installed
--> Processing Dependency: libnfnetlink.so.0()(64bit) for package:
iptables-1.4.21-18.3.el7_4.x86_64
--> Processing Dependency: libnetfilter_conntrack.so.3()(64bit) for package:
iptables-1.4.21-18.3.el7_4.x86_64
--> Running transaction check
---> Package libnetfilter_conntrack.x86_64 0:1.0.6-1.el7_3 will be installed
--> Processing Dependency: libmnl.so.0(LIBMNL_1.1)(64bit) for package:
libnetfilter_conntrack-1.0.6-1.el7_3.x86_64
--> Processing Dependency: libmnl.so.0(LIBMNL_1.0)(64bit) for package:
libnetfilter_conntrack-1.0.6-1.el7_3.x86_64

Complete!
[root@27df14c224be /]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=59 time=12.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=59 time=4.42 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=59 time=8.54 ms
^Z
[1]+  Stopped                 ping 8.8.8.8
[root@27df14c224be /]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
[root@27df14c224be /]# iptables -A OUTPUT -d 8.8.8.8 -j REJECT
[root@27df14c224be /]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
From 172.17.0.2 icmp_seq=1 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 172.17.0.2 icmp_seq=2 Destination Port Unreachable
ping: sendmsg: Operation not permitted
From 172.17.0.2 icmp_seq=3 Destination Port Unreachable
ping: sendmsg: Operation not permitted
^Z
```

OUTPUT SCREENS:

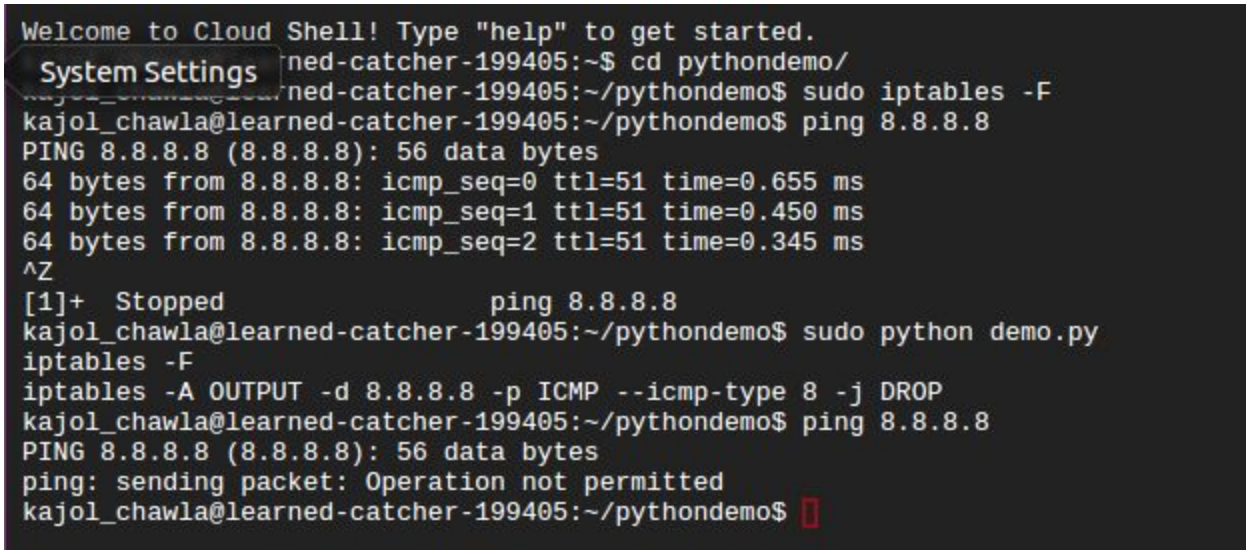
DockerFile:



sanj.py



GCP OUTPUT:



CENTOS OUTPUT:

```
[root@27df14c224be /]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=59 time=12.9 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=59 time=4.42 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=59 time=8.54 ms
^Z
[1]+  Stopped                 ping 8.8.8.8
[root@27df14c224be /]# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
```

CONCLUSION: A Docker container firewall should be a ‘must-have’ requirement before deploying any container-based applications. Thus, we created a container on docker after pulling centos library. Firewall rules were written inside the container. Thus, we were able to control the incoming and outgoing network traffic based on predetermined rules.