Software Defined Networks

Identify a “WHO IS” Solution

Team Members:

Priyank Shah

Meetika Sharma

Parth Shirolawala

Shubham Kothari

Goal: BlackListing IP's to make a Firewall system through a GRE tunnel between two SDN environment.

Environment Used:

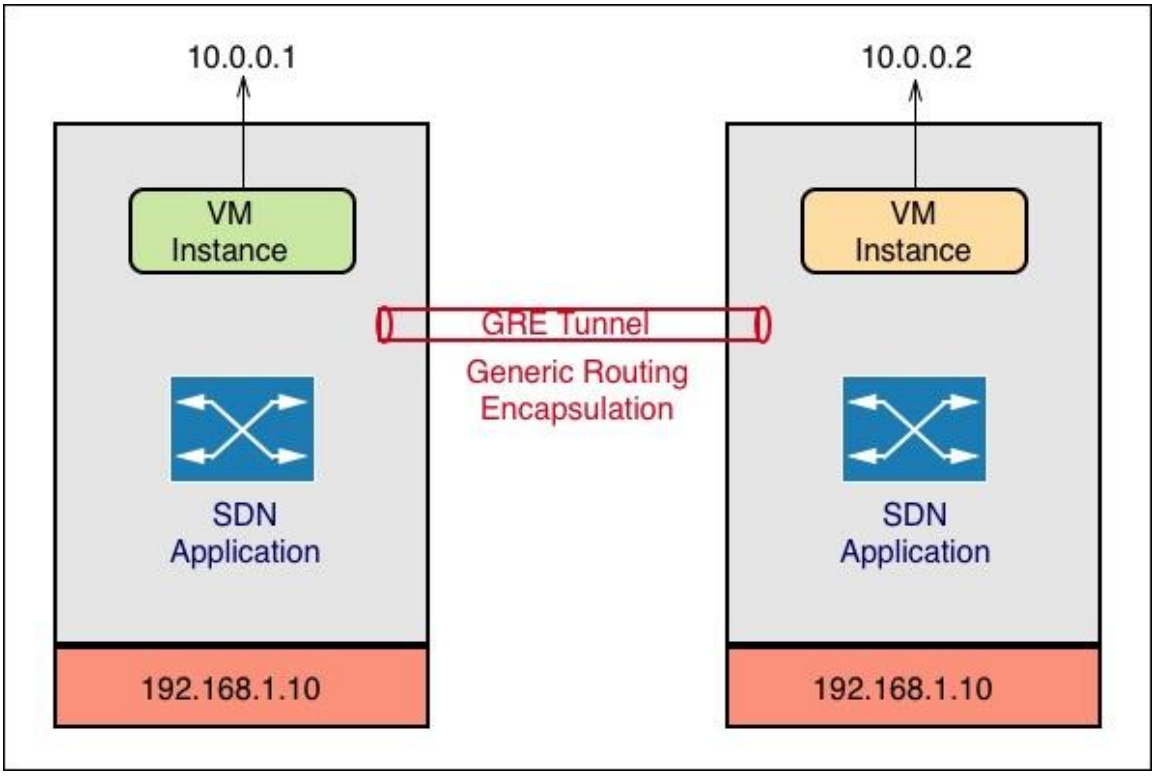
- Controller: POX Controller
- Topology: Linear Topology
- Tool: Mininet
- Language: Python

Features:

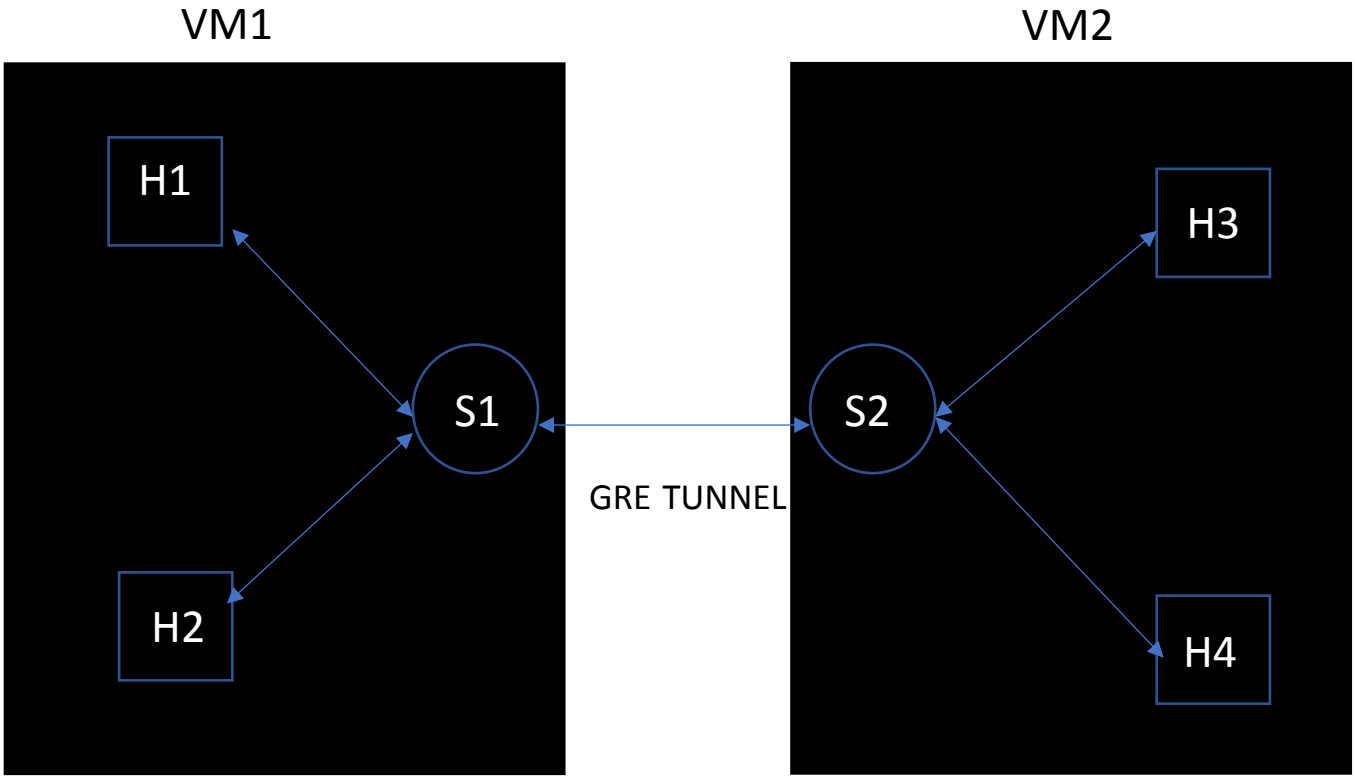
A Database is maintained for the registered domains listing. Data is used to identify and fix problems.

Basically Firewall is used to combat spam and fraud hosts from attacking a system. For accountability and for legal purposes.

Architecture:



Topology:



General Routing Encapsulation(GRE) is a tunneling protocol developed by Cisco Systems.

It Encapsulates packets to route other protocol over IP networks. It creates point to point connections between networks.

Implementation:

1) Run the Mininet VM instance

2) cd mininet

3) Create Tree.py with the following code

Mininet VM 1 - Tree.py

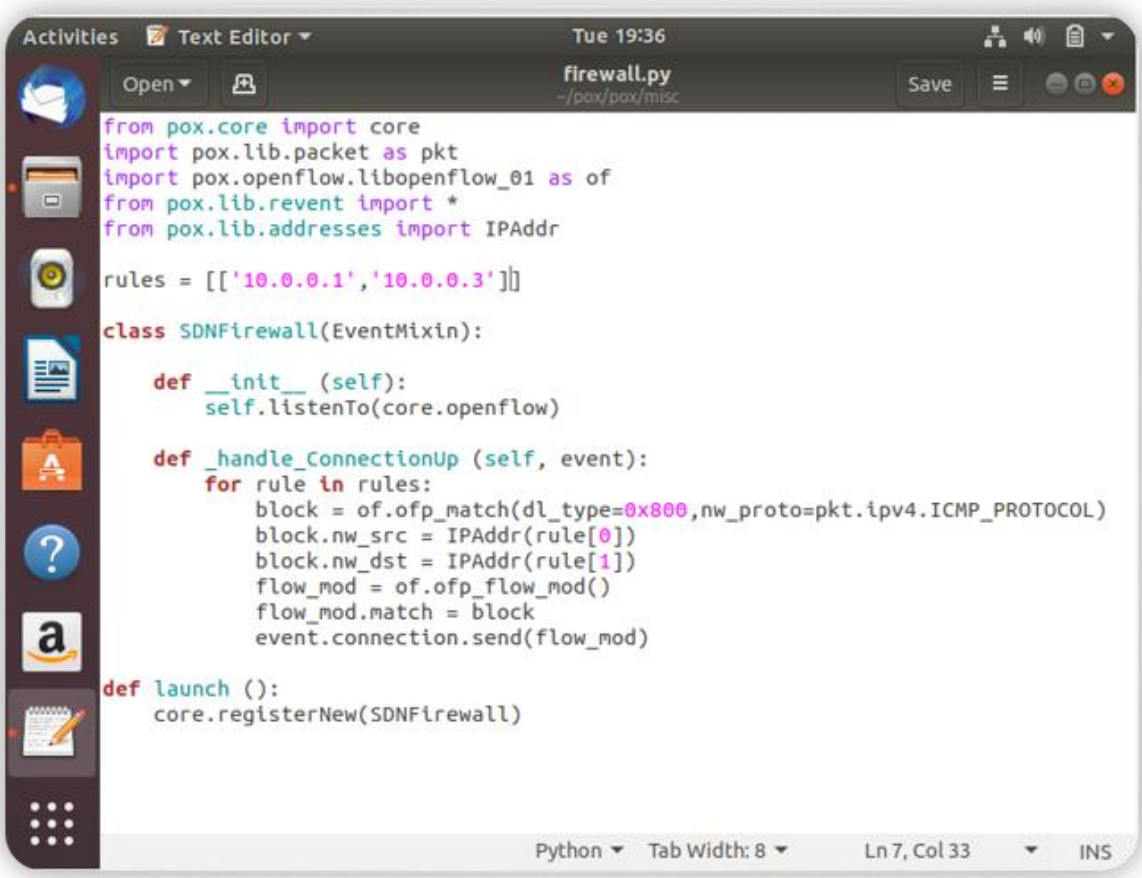
```
from mininet.topo import Topo

class treeTopo(Topo):
        def __init__(self):
                Topo.__init__(self)

                h1 = self.addHost('h1', ip='10.0.0.1')
                h2 = self.addHost('h2', ip='10.0.0.2')

                s1 = self.addSwitch('s1')

                self.addLink(h1, s1)
                self.addLink(h2, s1)

topos = {'mytopo':(lambda: treeTopo())}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"tree.py" 15L, 297C                                    1,1           All
```

Mininet VM 2 - Tree.py

```
from mininet.topo import Topo

class treeTopo(Topo):
        def __init__(self):
                Topo.__init__(self)

                h3 = self.addHost('h3',ip='10.0.0.3')
                h4 = self.addHost('h4',ip='10.0.0.4')

                s2 = self.addSwitch('s2')

                self.addLink(h3,s2)
                self.addLink(h4,s2)

topos = {'mytopo':(lambda: treeTopo())}
~
~
~
~
~
~
~
~
~
~
~
~
~
"tree.py" 15L, 294C                                    1,1           All
```

Steps to install POX controller with custom Firewall:

- Clone the git repository - git clone https://github.com/noxrepo/pox
- cd into the POX directory – cd pox/pox/misc
- Create/Update firewall.py with the following code
- Rules is a list of list which contains the IP addresses of the hosts to be blocked.
- We can add multiple rules to block multiple flows.



Steps to run the POX controller on Ubuntu:

- Get the IP address of the machine where POX controller will run using following command – ip a
- E.g. 192.168.0.41
- cd pox
- ./pox.py openflow.of_01 forwarding.l2_learning misc.firewall

```
meetika@meetika-VirtualBox:~/pox/pox/misc$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g
roup default qlen 1000
    link/ether 08:00:27:ae:47:3a brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.41/24 brd 192.168.0.255 scope global dynamic noprefixroute e
np0s3
```

```
meetika@meetika-VirtualBox:~/pox$ ./pox.py openflow.of_01 forwarding.l2_learnin
g misc.firewall
POX 0.5.0 (eel) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.5.0 (eel) is up.
INFO:openflow.of_01:[00-00-00-00-00-02 1] connected
```

Connect both Mininet VM to remote POX controller using the following command

1) cd mininet
2) sudo mn --custom Tree.py --topo mytopo --mac --controller=remote,ip=192.168.0.41,port=6633

Tree.py – filename of your topology

192.168.0.41 – IP address of the machine where POX controller is running

Create a GRE tunnel between mininets using the following command

On VM 1

sh ovs-vsctl add-port s1 hello -- set interface hello type=gre options:remote_ip=192.168.0.50

On VM 2

sh ovs-vsctl add-port s2 hello -- set interface hello type=gre options:remote_ip=192.168.0.51

s1, s2 – Name of the switch used in the topology

192.168.0.XX – IP address of the other mininet

- VM1 contains
  - 1 Switch (s1)
  - 2 Hosts (H1 (IP = 10.0.0.1),
  - H2 (IP = 10.0.0.2))

```
mininet> h1 ping -c 1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

- VM2 contains
  - 1 Switch (s2)
  - 2 Hosts (H3 (IP = 10.0.0.3),
  - H4 (IP = 10.0.0.4))

```
mininet> h1 ping -c 1 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=67.9 ms

--- 10.0.0.4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 67.915/67.915/67.915/0.000 ms
```

Flow from Host1 to Host3 is blocked.

Challenges Faced:

We blacklisted the IP, but we faced issues while doing reverse job i.e. Whitelisting the IP for certain hosts.

Configuring the GRE Tunnel.

Accessing mininet on UTD network.