

## Results of Firewall Implementation

I will discuss the problem and results of the solution implemented with the RYU controller to manage the network traffic. In this set up there are four networks attached to a singular switch constituting the patient, doctor, records and public access networks.

In this scenario the doctor network may exchange packets with the patient network and patient records however the patient and records networks may not communicate, and no one may communicate with the public access network.

```

vagrant@ubuntu-focal: /vagrant_data
[Switch 1]: PACKET-IN (#39993) on port: 3
[Switch 1]: PACKET-IN (#39994) on port: 3
[Switch 1]: PACKET-IN (#39995) on port: 3
[Switch 1]: PACKET-IN (#39996) on port: 3
[Switch 1]: PACKET-IN (#39997) on port: 3
[Switch 1]: PACKET-IN (#39998) on port: 3
[Switch 1]: PACKET-IN (#39999) on port: 3
[Switch 1]: PACKET-IN (#40000) on port: 3
[Switch 1]: PACKET-IN (#40001) on port: 3
[Switch 1]: PACKET-IN (#40002) on port: 3
[Switch 1]: PACKET-IN (#40003) on port: 3
[Switch 1]: PACKET-IN (#40004) on port: 3
[Switch 1]: PACKET-IN (#40005) on port: 3
[Switch 1]: PACKET-IN (#40006) on port: 3
[Switch 1]: PACKET-IN (#40007) on port: 3
[Switch 1]: PACKET-IN (#40008) on port: 3
[Switch 1]: PACKET-IN (#40009) on port: 3
[Switch 1]: PACKET-IN (#40010) on port: 3
[Switch 1]: PACKET-IN (#40011) on port: 3
[Switch 1]: PACKET-IN (#40012) on port: 3

Task: Secure Medical IOT Gateway Topology:
Scenario: firewall
Description:
Implement the following rules:
1. Patient Network (L1) may exchange packets with Doctor Network (H1) 2. Patient Records (R1) may exchange packets with Doctor Network (H1) 3. Noone from these three networks (L1,R1,H1) may send packets to Public Network Access (M1). 4. Patient Network (L1) and Patient Records (R1) may not exchange packets.

r1 -> X X X
m1 -> X X X
*** Results: 100% dropped (0/12 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X
l1 -> X X X
r1 -> X X X
m1 -> X X X
*** Results: 100% dropped (0/12 received)
mininet> net
h1 h1-eth0:s1-eth1
l1 l1-eth0:s1-eth2
r1 r1-eth0:s1-eth3
m1 m1-eth0:s1-eth4
s1 lo: s1-eth1:h1-eth0 s1-eth2:l1-eth0 s1-eth3:r1-eth0 s1-eth4:m1-eth0
c0
mininet> dpctl dump-flows

Error: 10000 packets expected at h1 (0 received)
Error: 5000 packets expected at l1 (0 received)
Error: 4000 packets expected at r1 (0 received)
Error: 0 packets expected at m1 (10000 received)

Status: FAILURE
sdn] 0:[tmux]*

```

Figure 1. A demonstration of the base application with all packets sent to the public access network.

The test involves sending traffic and counting the packets received at its destination as shown the base application fails to pass the test as all traffic is sent to m1. The controller has no other flows established to handle incoming traffic as such traffic is not effectively routed.

```

--- 33.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4079ms
rtt min/avg/max/mdev = 0.107/0.150/0.294/0.071 ms
mininet> l1 ping r1 -c 5
PING 33.0.0.1 (33.0.0.1) 56(84) bytes of data.

--- 33.0.0.1 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4367ms

mininet> m1 ping r1 -c 3
PING 33.0.0.1 (33.0.0.1) 56(84) bytes of data.

--- 33.0.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2041ms

mininet> m1 ping l1 -c 3
PING 22.0.0.1 (22.0.0.1) 56(84) bytes of data.

--- 22.0.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2050ms

mininet> m1 ping h1 -c 3
PING 11.0.0.1 (11.0.0.1) 56(84) bytes of data.

--- 11.0.0.1 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2051ms

mininet>

```

Figure 2. The RYU controller logic unable to effectively direct ICMP traffic.

```

vagrant@ubuntu-focal: /vagrant_data
[Switch 1]: PACKET-IN (#30983) on port: 1
[Switch 1]: PACKET-IN (#30984) on port: 1
[Switch 1]: PACKET-IN (#30985) on port: 1
[Switch 1]: PACKET-IN (#30986) on port: 1
[Switch 1]: PACKET-IN (#30987) on port: 1
[Switch 1]: PACKET-IN (#30988) on port: 1
[Switch 1]: PACKET-IN (#30989) on port: 1
[Switch 1]: PACKET-IN (#30990) on port: 1
[Switch 1]: PACKET-IN (#30991) on port: 1
[Switch 1]: PACKET-IN (#30992) on port: 1
[Switch 1]: PACKET-IN (#30993) on port: 1
[Switch 1]: PACKET-IN (#30994) on port: 1
[Switch 1]: PACKET-IN (#30995) on port: 1
[Switch 1]: PACKET-IN (#30996) on port: 1
[Switch 1]: PACKET-IN (#30997) on port: 1
[Switch 1]: PACKET-IN (#30998) on port: 1
[Switch 1]: PACKET-IN (#30999) on port: 1
[Switch 1]: PACKET-IN (#31000) on port: 1
[Switch 1]: PACKET-IN (#31001) on port: 2
[Switch 1]: PACKET-IN (#31002) on port: 4
[Switch 1]: PACKET-IN (#31003) on port: 1
[Switch 1]: PACKET-IN (#31004) on port: 3

Task: Secure Medical IOT Gateway Topology:
Scenario: firewall
Description:
Implement the following rules:
1. Patient Network (L1) may
exchange packets with Doctor
Network (H1) 2. Patient
Records (R1) may exchange
packets with Doctor Network
(H1) 3. Noone from these three
networks (L1,R1,H1) may send
packets to Public Network
Access (M1). 4. Patient
Network (L1) and Patient
Records (R1) may not exchange
packets.
Result: SUCCESS

mininet>
mininet> net
h1 h1-eth0:s1-eth1
l1 l1-eth0:s1-eth2
r1 r1-eth0:s1-eth3
m1 m1-eth0:s1-eth4
s1 lo: s1-eth1:h1-eth0 s1-eth2:l1-eth0 s1-eth3:r1-eth0 s1-eth4:m1-eth0
c0
mininet>
mininet> links
h1-eth0<=>s1-eth1 (OK OK)
l1-eth0<=>s1-eth2 (OK OK)
r1-eth0<=>s1-eth3 (OK OK)
m1-eth0<=>s1-eth4 (OK OK)
mininet>
mininet> pingall
*** Ping: testing ping reachability
h1 -> l1 r1 X
l1 -> h1 X X
r1 -> h1 X X
m1 -> X X X
*** Results: 66% dropped (4/12 received)
mininet>
mininet> dpctl dump-flows
*** s1
-
cookie=0x0, duration=137.457s, table=0, n_packets=26, n_bytes=1092, priority=
1,arp actions=FLOOD
cookie=0x0, duration=137.457s, table=0, n_packets=2, n_bytes=196, priority=1,
ip,nw_src=11.0.0.0/8,nw_dst=22.0.0.0/8 actions=output:"s1-eth2"
cookie=0x0, duration=137.457s, table=0, n_packets=2, n_bytes=196, priority=1,
ip,nw_src=22.0.0.0/8,nw_dst=11.0.0.0/8 actions=output:"s1-eth1"
cookie=0x0, duration=137.457s, table=0, n_packets=2, n_bytes=196, priority=1,
ip,nw_src=11.0.0.0/8,nw_dst=33.0.0.0/8 actions=output:"s1-eth3"
cookie=0x0, duration=137.458s, table=0, n_packets=2, n_bytes=196, priority=1,
ip,nw_src=33.0.0.0/8,nw_dst=11.0.0.0/8 actions=output:"s1-eth1"
cookie=0x0, duration=137.458s, table=0, n_packets=50, n_bytes=4012, priority=
0 actions=CONTROLLER:65509
mininet>

```

Figure 3. The solution implemented and the results when the traffic scenario is instantiated.

When the firewall.py application is run in the RYU controller It successfully passes the scenario that the unaltered application did not.

This was achieved by creating flow roles based upon incoming packet ipv4\_src and ipv4\_dst.

Below in figure 4. The pingall command demonstrates a reachability that matches the plan document and the flow roles which have been instigated.

```

vagrant@ubuntu-focal: /vagrant_data$ sudo python3 remote/script_run_mininet.py
/vagrant_data/local/apps/scenarios/demo.yaml
mininet> exit
vagrant@ubuntu-focal: /vagrant_data$ sudo python3 remote/script_run_mininet.py
/vagrant_data/local/apps/scenarios/project/firewall.yaml
mininet> net
h1 h1-eth0:s1-eth1
l1 l1-eth0:s1-eth2
r1 r1-eth0:s1-eth3
m1 m1-eth0:s1-eth4
s1 lo: s1-eth1:h1-eth0 s1-eth2:l1-eth0 s1-eth3:r1-eth0 s1-eth4:m1-eth0
c0
mininet> pingall
*** Ping: testing ping reachability
h1 -> l1 r1 X
l1 -> h1 X X
r1 -> h1 X X
m1 -> X X X
*** Results: 66% dropped (4/12 received)
mininet>
mininet> dpctl dump-flows
*** s1
-
cookie=0x0, duration=137.457s, table=0, n_packets=26, n_bytes=1092, priority=
1,arp actions=FLOOD
cookie=0x0, duration=137.457s, table=0, n_packets=2, n_bytes=196, priority=1,
ip,nw_src=11.0.0.0/8,nw_dst=22.0.0.0/8 actions=output:"s1-eth2"
cookie=0x0, duration=137.457s, table=0, n_packets=2, n_bytes=196, priority=1,
ip,nw_src=22.0.0.0/8,nw_dst=11.0.0.0/8 actions=output:"s1-eth1"
cookie=0x0, duration=137.457s, table=0, n_packets=2, n_bytes=196, priority=1,
ip,nw_src=11.0.0.0/8,nw_dst=33.0.0.0/8 actions=output:"s1-eth3"
cookie=0x0, duration=137.458s, table=0, n_packets=2, n_bytes=196, priority=1,
ip,nw_src=33.0.0.0/8,nw_dst=11.0.0.0/8 actions=output:"s1-eth1"
cookie=0x0, duration=137.458s, table=0, n_packets=50, n_bytes=4012, priority=
0 actions=CONTROLLER:65509
mininet>

```

Figure 4. Demonstration of switch connectivity and the flows implemented.



```

--- 22.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4100ms
rtt min/avg/max/mdev = 0.064/0.148/0.287/0.077 ms
mininet> h1 ping l1 -c 5
PING 22.0.0.1 (22.0.0.1) 56(84) bytes of data.
64 bytes from 22.0.0.1: icmp_seq=1 ttl=64 time=0.209 ms
64 bytes from 22.0.0.1: icmp_seq=2 ttl=64 time=0.036 ms
64 bytes from 22.0.0.1: icmp_seq=3 ttl=64 time=0.058 ms
64 bytes from 22.0.0.1: icmp_seq=4 ttl=64 time=0.105 ms
64 bytes from 22.0.0.1: icmp_seq=5 ttl=64 time=0.103 ms

--- 22.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4085ms
rtt min/avg/max/mdev = 0.036/0.102/0.209/0.059 ms
mininet> h1 ping r1 -c 5
PING 33.0.0.1 (33.0.0.1) 56(84) bytes of data.
64 bytes from 33.0.0.1: icmp_seq=1 ttl=64 time=0.294 ms
64 bytes from 33.0.0.1: icmp_seq=2 ttl=64 time=0.118 ms
64 bytes from 33.0.0.1: icmp_seq=3 ttl=64 time=0.116 ms
64 bytes from 33.0.0.1: icmp_seq=4 ttl=64 time=0.118 ms
64 bytes from 33.0.0.1: icmp_seq=5 ttl=64 time=0.107 ms

--- 33.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4079ms
rtt min/avg/max/mdev = 0.107/0.150/0.294/0.071 ms

```

*Figure 5. Depicts reachability between the doctor-patient and doctor-records networks*

ICMP test determines reachability by sending echo request packets and await the echoed response from the target host. Various statistics can be derived from this such as packet loss percentage, time from message sent to response receival including the fastest, slowest, packet average and standard deviation from the mean.