Universidade de Aveiro

Segurança em Redes de Comunicações

Report of Project 1

André Clérigo (98485), Pedro Rocha (98256)

Departamento de Eletrónica, Telecomunicações e Informática

April 16th, 2023

Contents

List of Figures

Chapter 1

Topology and configurations

1.1 Overview

We structured a network composed of three zones: INSIDE, OUTSIDE and DMZ. In the INSIDE zone (internal network), there are two subnetworks. The 10.10.10.0/24 network, which we named the Admin Network, contains a VPCS with the IP address 10.10.10.2. The remaining IPs of 10.0.0.0/8 network is designated for the rest of the company terminals.

The OUTSIDE zone (Internet) consists of the 200.2.2.0/24 subnetwork with a VPCS to simulate an external device that we want to access from a company terminal.

Lastly, the DMZ zone includes the 192.1.1.0/24 subnetwork with three IP addresses. The 192.1.1.100 and 192.1.1.200 IPs are part of the DMZ-Server, the 192.1.1.200 simulates a web service accessible both internally and externally, 192.1.1.100 simulates another web service accessible only internally, which can represent an internal company web service such as an admin page. Additionally, we have instantiated a VPCS with the IP address 192.1.1.150 to simulate a DNS server.

This topology is depicted bellow with some other aspects of the network, in Figure 1.1.

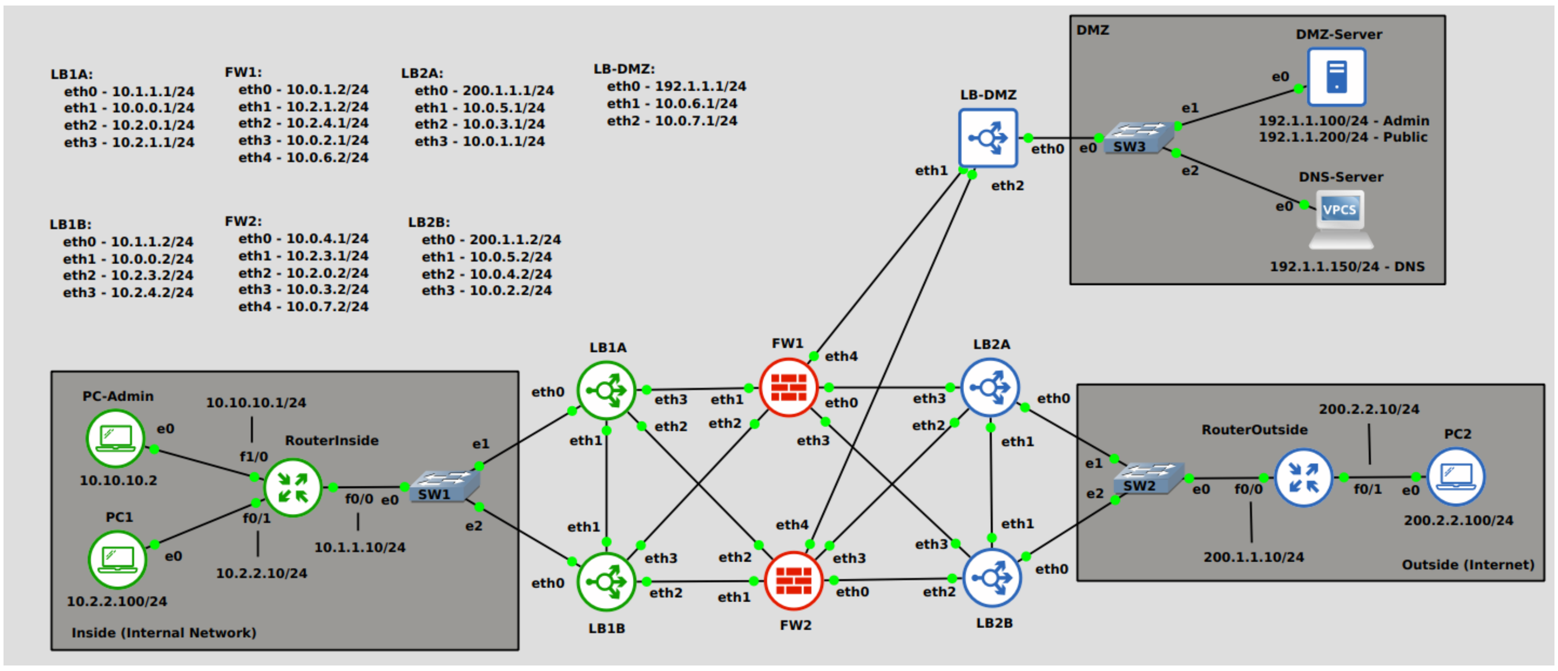


Figure 1.1: Topology of the network.

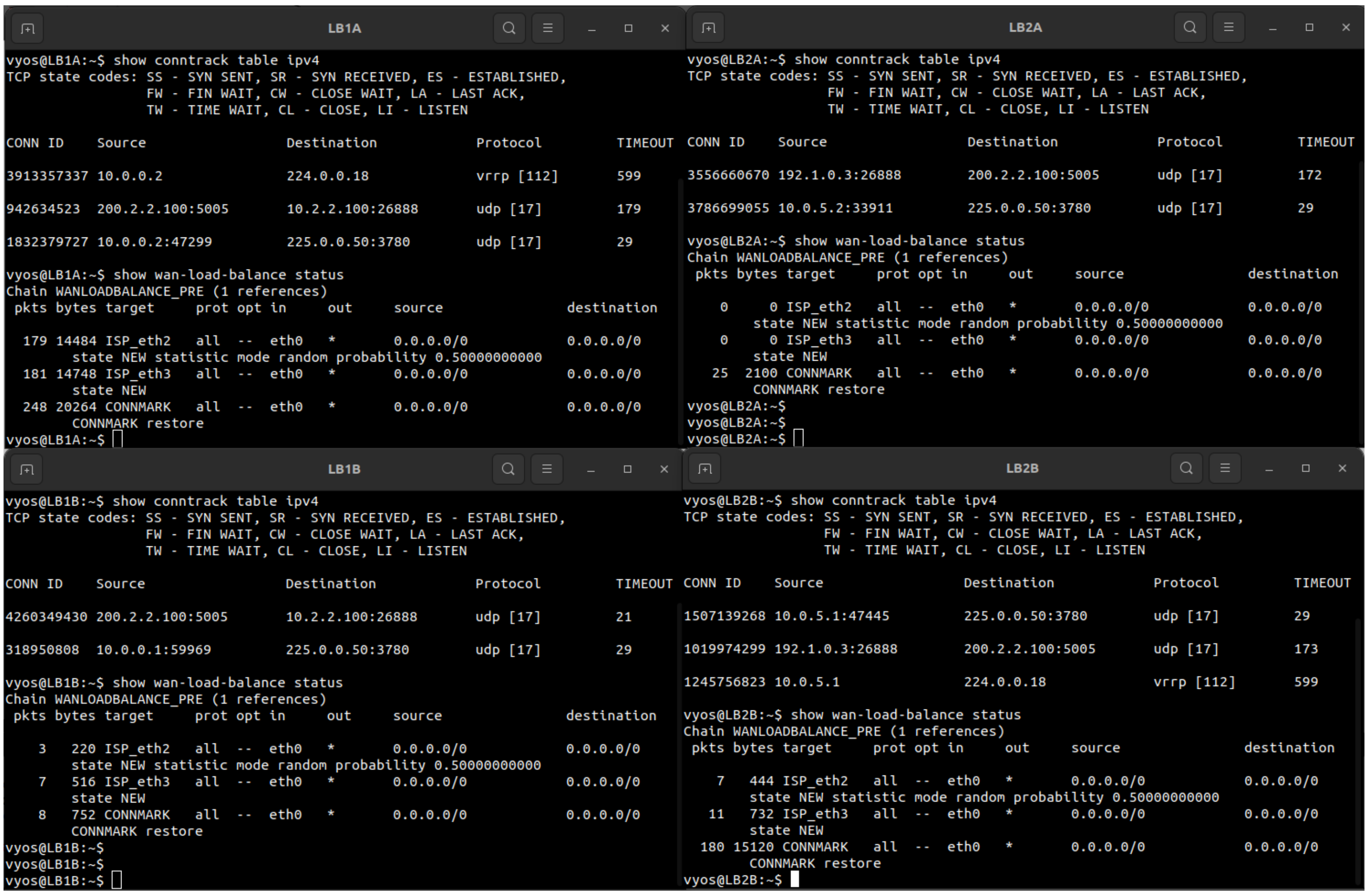
1.2 Configurations

1.2.1 INSIDE and OUTSIDE

In the INSIDE and OUTSIDE networks, our configurations consisted in assigning the correct gateways to the VPCS (the ones directly connected and with the same subnetwork). On the RouterInside, we defined static routes for LB1A and LB1B, while on the RouterOutside, we set up static routes for LB2A and LB2B.

1.2.2 LBs and FWs

In the Load Balancers (LBs), we created static routes for the edge. For LB1A and LB1B, we set up static routes forwarding the 10.10.10.0/24 and 10.2.2.0/24 networks to the RouterInside interface (10.1.1.10). In contrast, for LB2A and LB2B, we established static routes directing traffic to the 200.2.2.0/24 network through the RouterOutside interface (200.1.1.10). This static routes configuration can be seen in lines 16 and 17 of Subsection A.1.1 and A.1.2. Additionally, all LBs have conntrack-sync configured on eth1, and load balancing on their eth2 and eth3 interfaces with equal weights, this means that the probability of forwarding a packet to an interface is 50% this is verified in Figure 1.2.



```
vyos@LB1A:~$ show conntrack table ipv4
TCP state codes: SS - SYN SENT, SR - SYN RECEIVED, ES - ESTABLISHED,
                 FW - FIN WAIT, CW - CLOSE WAIT, LA - LAST ACK,
                 TW - TIME WAIT, CL - CLOSE, LI - LISTEN

CONN ID    Source                 Destination            Protocol         TIMEOUT

3913357337 10.0.0.2               224.0.0.18             vrrp [112]       599

942634523  200.2.2.100:5005       10.2.2.100:26888       udp [17]         179

1832379727 10.0.0.2:47299         225.0.0.50:3780        udp [17]         29

vyos@LB1A:~$ show wan-load-balance status
Chain WANLOADBALANCE_PRE (1 references)
 pkts bytes target     prot opt in     out     source               destination

  179 14484 ISP_eth2   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state NEW statistic mode random probability 0.50000000000
  181 14748 ISP_eth3   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state NEW
  248 20264 CONNMARK   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        CONNMARK restore
vyos@LB1A:~$
```

```
vyos@LB2A:~$ show conntrack table ipv4
TCP state codes: SS - SYN SENT, SR - SYN RECEIVED, ES - ESTABLISHED,
                 FW - FIN WAIT, CW - CLOSE WAIT, LA - LAST ACK,
                 TW - TIME WAIT, CL - CLOSE, LI - LISTEN

CONN ID    Source                 Destination            Protocol         TIMEOUT

3556660670 192.1.0.3:26888        200.2.2.100:5005       udp [17]         172

3786699055 10.0.5.2:33911         225.0.0.50:3780        udp [17]         29

vyos@LB2A:~$ show wan-load-balance status
Chain WANLOADBALANCE_PRE (1 references)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ISP_eth2   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state NEW statistic mode random probability 0.50000000000
    0     0 ISP_eth3   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state NEW
   25  2100 CONNMARK   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        CONNMARK restore
vyos@LB2A:~$
vyos@LB2A:~$
vyos@LB2A:~$
```

```
vyos@LB1B:~$ show conntrack table ipv4
TCP state codes: SS - SYN SENT, SR - SYN RECEIVED, ES - ESTABLISHED,
                 FW - FIN WAIT, CW - CLOSE WAIT, LA - LAST ACK,
                 TW - TIME WAIT, CL - CLOSE, LI - LISTEN

CONN ID    Source                 Destination            Protocol         TIMEOUT

4260349430 200.2.2.100:5005       10.2.2.100:26888       udp [17]         21

318950808  10.0.0.1:59969         225.0.0.50:3780        udp [17]         29

vyos@LB1B:~$ show wan-load-balance status
Chain WANLOADBALANCE_PRE (1 references)
 pkts bytes target     prot opt in     out     source               destination

    3   220 ISP_eth2   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state NEW statistic mode random probability 0.50000000000
    7   516 ISP_eth3   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state NEW
    8   752 CONNMARK   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        CONNMARK restore
vyos@LB1B:~$
vyos@LB1B:~$
vyos@LB1B:~$
```

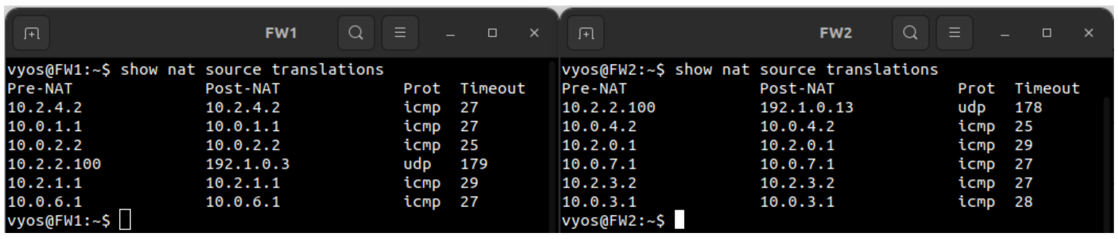
```
vyos@LB2B:~$ show conntrack table ipv4
TCP state codes: SS - SYN SENT, SR - SYN RECEIVED, ES - ESTABLISHED,
                 FW - FIN WAIT, CW - CLOSE WAIT, LA - LAST ACK,
                 TW - TIME WAIT, CL - CLOSE, LI - LISTEN

CONN ID    Source                 Destination            Protocol         TIMEOUT

1507139268 10.0.5.1:47445         225.0.0.50:3780        udp [17]         29

1019974299 192.1.0.3:26888        200.2.2.100:5005       udp [17]         173

1245756823 10.0.5.1               224.0.0.18             vrrp [112]       599

vyos@LB2B:~$ show wan-load-balance status
Chain WANLOADBALANCE_PRE (1 references)
 pkts bytes target     prot opt in     out     source               destination

    7   444 ISP_eth2   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state NEW statistic mode random probability 0.50000000000
   11   732 ISP_eth3   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        state NEW
  180 15120 CONNMARK   all  --  eth0   *       0.0.0.0/0            0.0.0.0/0
        CONNMARK restore
vyos@LB2B:~$
```

Figure 1.2: Load Balancers Synchronization and Probability.

For the Firewalls (FWs), we configured static routes from INSIDE to OUTSIDE and from OUTSIDE to INSIDE. Traffic heading to the 10.0.0.0/8 network is directed through the next-hops connected to the eth1 and eth2 interfaces, while traffic destined for the 200.2.2.0/24 network is routed through the next-hops linked to the eth0 and eth3 interfaces. This static routes configurations can be seen in lines 62 to 66 of Subsection B.1. We also activated the SSH service on port 22, gave access-control to the default user (vyos), and added a little bit of security by setting listen-address to the interface eth4, so that only devices connected to that interface would be able to access it through SSH, these configurations can be seen in lines 68 to 70 of Section B.1.

One important consideration is that the NAT pool between FWs must be distinct because there is a minimal chance that both FWs might select the same IP:PORT translation for different internal IPs, which is why we decided to separate the pools. This NAT configuration can be seen in lines 54 to 60 of Section B.1 and lines 3 to 9 of Section B.2. FW1 utilizes a pool ranging from 192.168.1.0 to 192.168.1.10, while FW2 uses a range from 192.168.1.11 to 192.168.1.20, this can be seen by the translations done by the firewall seen in Figure 1.3.

Another aspect to consider is that the LB2A and LB2B must have matching interfaces connected to the FWs. This is because the conntrack-sync records the packet that passed through the LBs and the interface on which it was received. Therefore, both LBs must have the same interface connected to the same FW, ensuring that if a request is handled by LB2B, the reply coming through LB2A can be redirected to the



FW1

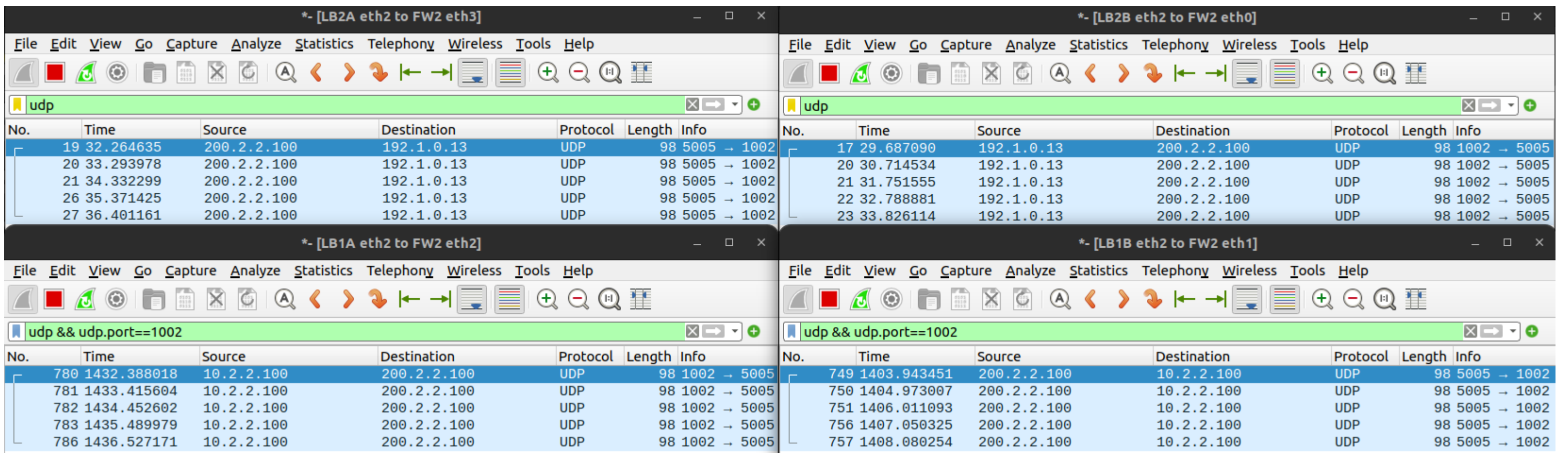
```
vyos@FW1:~$ show nat source translations
Pre-NAT                   Post-NAT                  Prot  Timeout
10.2.4.2                  10.2.4.2                  icmp  27
10.0.1.1                  10.0.1.1                  icmp  27
10.0.2.2                  10.0.2.2                  icmp  25
10.2.2.100                192.1.0.3                 udp   179
10.2.1.1                  10.2.1.1                  icmp  29
10.0.6.1                  10.0.6.1                  icmp  27
vyos@FW1:~$
```

FW2

```
vyos@FW2:~$ show nat source translations
Pre-NAT                   Post-NAT                  Prot  Timeout
10.2.2.100                192.1.0.13                udp   178
10.0.4.2                  10.0.4.2                  icmp  25
10.2.0.1                  10.2.0.1                  icmp  29
10.0.7.1                  10.0.7.1                  icmp  27
10.2.3.2                  10.2.3.2                  icmp  27
10.0.3.1                  10.0.3.1                  icmp  28
vyos@FW2:~$
```

Figure 1.3: NAT Translations.

correct FW, depicted in Figure 1.4. For LB1A and LB1B, this is not fully necessary because it does not matter which FW the request and reply go through, as there are no additional FWs between these LBs and the INSIDE network and there isn't traffic coming from the other zones to the INSIDE zone that aren't replies. The conntrack is configured in lines 19 to 23 of Subsections A.1.1 and A.1.2, while it's configured in lines 18 to 22 of Subsections A.2.1 and A.2.2.



*- [LB2A eth2 to FW2 eth3]

Filter: udp

```
No.   Time         Source        Destination   Protocol  Length  Info
   19 32.264635    200.2.2.100   192.1.0.13    UDP           98  5005 → 1002
   20 33.293978    200.2.2.100   192.1.0.13    UDP           98  5005 → 1002
   21 34.332299    200.2.2.100   192.1.0.13    UDP           98  5005 → 1002
   26 35.371425    200.2.2.100   192.1.0.13    UDP           98  5005 → 1002
   27 36.401161    200.2.2.100   192.1.0.13    UDP           98  5005 → 1002
```

*- [LB2B eth2 to FW2 eth0]

Filter: udp

```
No.   Time         Source        Destination   Protocol  Length  Info
   17 29.687090    192.1.0.13    200.2.2.100   UDP           98  1002 → 5005
   20 30.714534    192.1.0.13    200.2.2.100   UDP           98  1002 → 5005
   21 31.751555    192.1.0.13    200.2.2.100   UDP           98  1002 → 5005
   22 32.788881    192.1.0.13    200.2.2.100   UDP           98  1002 → 5005
   23 33.826114    192.1.0.13    200.2.2.100   UDP           98  1002 → 5005
```

*- [LB1A eth2 to FW2 eth2]

Filter: udp && udp.port==1002

```
No.   Time           Source        Destination   Protocol  Length  Info
  780 1432.388018    10.2.2.100    200.2.2.100   UDP           98  1002 → 5005
  781 1433.415604    10.2.2.100    200.2.2.100   UDP           98  1002 → 5005
  782 1434.452602    10.2.2.100    200.2.2.100   UDP           98  1002 → 5005
  783 1435.489979    10.2.2.100    200.2.2.100   UDP           98  1002 → 5005
  786 1436.527171    10.2.2.100    200.2.2.100   UDP           98  1002 → 5005
```

*- [LB1B eth2 to FW2 eth1]

Filter: udp && udp.port==1002

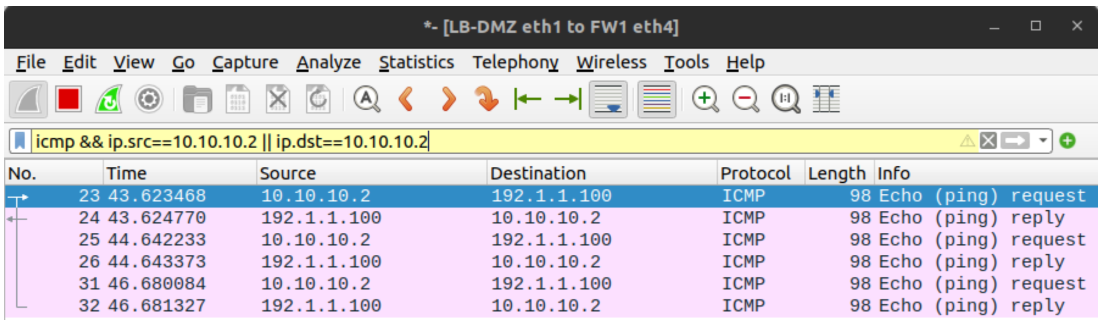
```
No.   Time           Source        Destination   Protocol  Length  Info
  749 1403.943451    200.2.2.100   10.2.2.100    UDP           98  5005 → 1002
  750 1404.973007    200.2.2.100   10.2.2.100    UDP           98  5005 → 1002
  751 1406.011093    200.2.2.100   10.2.2.100    UDP           98  5005 → 1002
  756 1407.050325    200.2.2.100   10.2.2.100    UDP           98  5005 → 1002
  757 1408.080254    200.2.2.100   10.2.2.100    UDP           98  5005 → 1002
```

Figure 1.4: Synchronization between Load Balancers and interface agreement.

It is important to note that in this case, the FWs do not require synchronization between them because the LBs are already synchronized. With the LBs synchronized, they can efficiently forward a reply to the FW that allowed the request to pass through. This ensures that the FWs do not need to be aware of each other's state.

1.2.3 DMZ

We added a LB in the DMZ because both FW1 and FW2 can send traffic to the DMZ. If we did not have an LB, we would need to use two gateways or a Router with static routes, which could cause problems since a request could come from one FW and the reply be sent to another, leading to connection issues. By introducing an LB connected to both FWs and the DMZ-Server and utilizing load-balancing and sticky connections, we can ensure that the reply always goes back to the FW that sent the request, as we can see in Figure 1.5. The DMZ's load balancer configuration can be found in Subsection A.3.



*- [LB-DMZ eth1 to FW1 eth4]

Filter: icmp && ip.src==10.10.10.2 || ip.dst==10.10.10.2

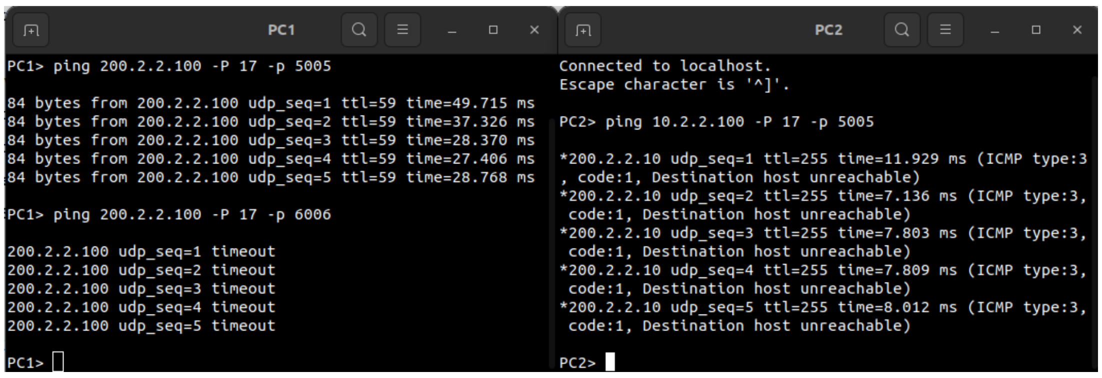
```
No.   Time         Source        Destination   Protocol  Length  Info
   23 43.623468    10.10.10.2    192.1.1.100   ICMP          98  Echo (ping) request
   24 43.624770    192.1.1.100   10.10.10.2    ICMP          98  Echo (ping) reply
   25 44.642233    10.10.10.2    192.1.1.100   ICMP          98  Echo (ping) request
   26 44.643373    192.1.1.100   10.10.10.2    ICMP          98  Echo (ping) reply
   31 46.680084    10.10.10.2    192.1.1.100   ICMP          98  Echo (ping) request
   32 46.681327    192.1.1.100   10.10.10.2    ICMP          98  Echo (ping) reply
```

Figure 1.5: Sticky Connections for LB-DMZ.

Chapter 2

Services

We defined several services for our network. The INSIDE network can communicate with the OUTSIDE using UDP on ports 5000 to 6000. This service simulates a scenario where devices in the INSIDE need to communicate with devices in the OUTSIDE through a video conferencing system that uses UDP, for example. In this situation, the OUTSIDE network cannot initiate the communication, as we can see on Figure 2.1.

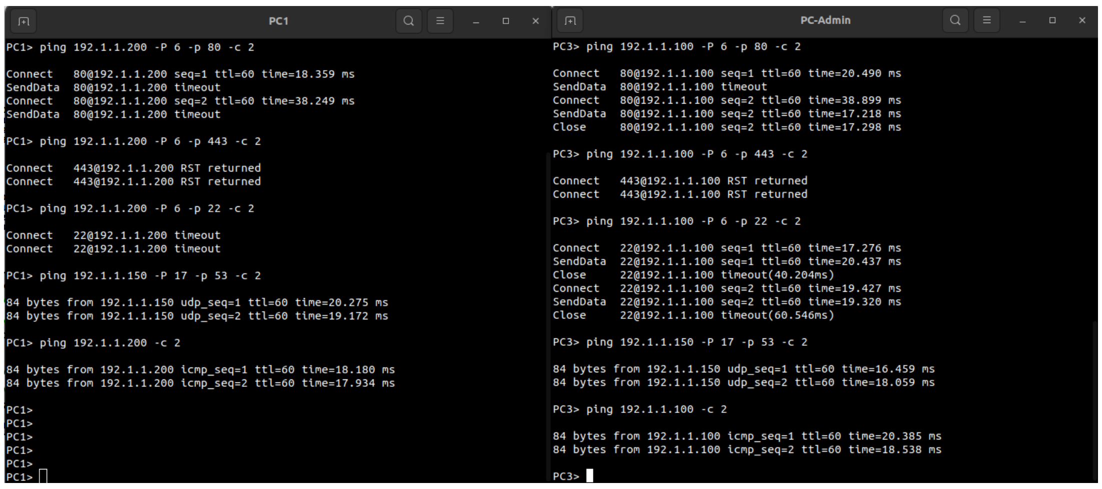


```
PC1> ping 200.2.2.100 -P 17 -p 5005

84 bytes from 200.2.2.100 udp_seq=1 ttl=59 time=49.715 ms
84 bytes from 200.2.2.100 udp_seq=2 ttl=59 time=37.326 ms
84 bytes from 200.2.2.100 udp_seq=3 ttl=59 time=28.370 ms
84 bytes from 200.2.2.100 udp_seq=4 ttl=59 time=27.406 ms
84 bytes from 200.2.2.100 udp_seq=5 ttl=59 time=28.768 ms

PC1> ping 200.2.2.100 -P 17 -p 6006

200.2.2.100 udp_seq=1 timeout
200.2.2.100 udp_seq=2 timeout
200.2.2.100 udp_seq=3 timeout
200.2.2.100 udp_seq=4 timeout
200.2.2.100 udp_seq=5 timeout

PC1>
```

```
Connected to localhost.
Escape character is '^]'.

PC2> ping 10.2.2.100 -P 17 -p 5005

*200.2.2.10 udp_seq=1 ttl=255 time=11.929 ms (ICMP type:3
, code:1, Destination host unreachable)
*200.2.2.10 udp_seq=2 ttl=255 time=7.136 ms (ICMP type:3,
 code:1, Destination host unreachable)
*200.2.2.10 udp_seq=3 ttl=255 time=7.803 ms (ICMP type:3,
 code:1, Destination host unreachable)
*200.2.2.10 udp_seq=4 ttl=255 time=7.809 ms (ICMP type:3,
 code:1, Destination host unreachable)
*200.2.2.10 udp_seq=5 ttl=255 time=8.012 ms (ICMP type:3,
 code:1, Destination host unreachable)

PC2>
```

Figure 2.1: Rules of INSIDE to OUTSIDE communication.

Another service we considered was INSIDE access to the DMZ. All internal devices can ping devices in the DMZ network, communicate over UDP on port 53 (to simulate a DNS communication), and also communicate via TCP on ports 80 and 443, enabling them to access web services over HTTP and HTTPS. Additionally, only the admin network (10.10.10.0/24) can communicate with the DMZ over TCP on port 22, meaning that only administrators can establish SSH connections to the terminals within the DMZ. This can be seen in Figure 2.2

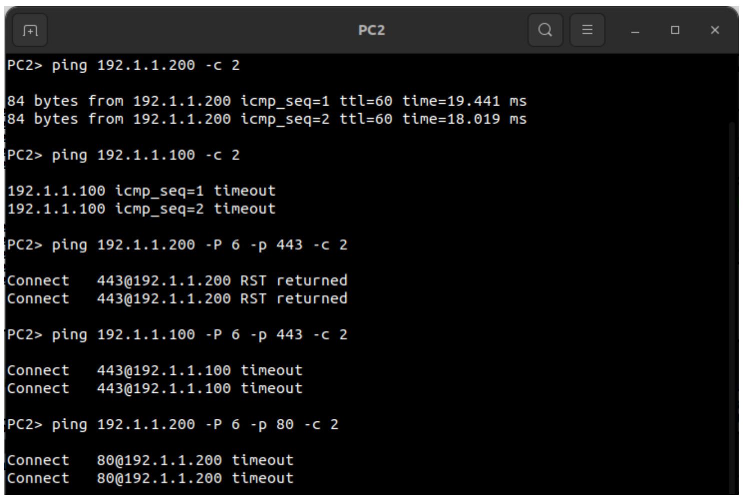


```
PC1> ping 192.1.1.200 -P 6 -p 80 -c 2

Connect   80@192.1.1.200 seq=1 ttl=60 time=18.359 ms
SendData  80@192.1.1.200 timeout
Connect   80@192.1.1.200 seq=2 ttl=60 time=38.249 ms
SendData  80@192.1.1.200 timeout

PC1> ping 192.1.1.200 -P 6 -p 443 -c 2

Connect   443@192.1.1.200 RST returned
Connect   443@192.1.1.200 RST returned

PC1> ping 192.1.1.200 -P 6 -p 22 -c 2

Connect   22@192.1.1.200 timeout
Connect   22@192.1.1.200 timeout

PC1> ping 192.1.1.150 -P 17 -p 53 -c 2

84 bytes from 192.1.1.150 udp_seq=1 ttl=60 time=20.275 ms
84 bytes from 192.1.1.150 udp_seq=2 ttl=60 time=19.172 ms

PC1> ping 192.1.1.200 -c 2

84 bytes from 192.1.1.200 icmp_seq=1 ttl=60 time=18.180 ms
84 bytes from 192.1.1.200 icmp_seq=2 ttl=60 time=17.934 ms

PC1>
PC1>
PC1>
PC1>
PC1>
PC1>
```

```
PC3> ping 192.1.1.100 -P 6 -p 80 -c 2

Connect   80@192.1.1.100 seq=1 ttl=60 time=20.490 ms
SendData  80@192.1.1.100 timeout
Connect   80@192.1.1.100 seq=2 ttl=60 time=38.899 ms
SendData  80@192.1.1.100 seq=2 ttl=60 time=17.218 ms
Close     80@192.1.1.100 seq=2 ttl=60 time=17.298 ms

PC3> ping 192.1.1.100 -P 6 -p 443 -c 2

Connect   443@192.1.1.100 RST returned
Connect   443@192.1.1.100 RST returned

PC3> ping 192.1.1.100 -P 6 -p 22 -c 2

Connect   22@192.1.1.100 seq=1 ttl=60 time=17.276 ms
SendData  22@192.1.1.100 seq=1 ttl=60 time=20.437 ms
Close     22@192.1.1.100 timeout(40.204ms)
Connect   22@192.1.1.100 seq=2 ttl=60 time=19.427 ms
SendData  22@192.1.1.100 seq=2 ttl=60 time=19.320 ms
Close     22@192.1.1.100 timeout(60.546ms)

PC3> ping 192.1.1.150 -P 17 -p 53 -c 2

84 bytes from 192.1.1.150 udp_seq=1 ttl=60 time=16.459 ms
84 bytes from 192.1.1.150 udp_seq=2 ttl=60 time=18.059 ms

PC3> ping 192.1.1.100 -c 2

84 bytes from 192.1.1.100 icmp_seq=1 ttl=60 time=20.385 ms
84 bytes from 192.1.1.100 icmp_seq=2 ttl=60 time=18.538 ms

PC3>
```

Figure 2.2: Rules of INSIDE to DMZ communication.

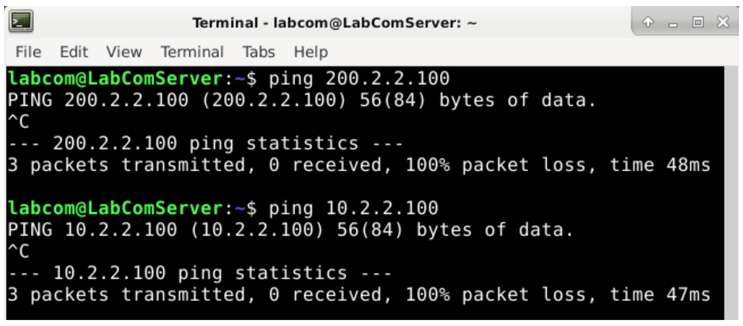
For the OUTSIDE network, we defined a service in which devices can only communicate with the IP address 192.1.1.200 (public page). The allowed communication methods are through pings or via TCP on port 443, which enables access over HTTPS, this behaviour can be seen in Figure 2.3.



```
PC2> ping 192.1.1.200 -c 2

84 bytes from 192.1.1.200 icmp_seq=1 ttl=60 time=19.441 ms
84 bytes from 192.1.1.200 icmp_seq=2 ttl=60 time=18.019 ms

PC2> ping 192.1.1.100 -c 2

192.1.1.100 icmp_seq=1 timeout
192.1.1.100 icmp_seq=2 timeout

PC2> ping 192.1.1.200 -P 6 -p 443 -c 2

Connect   443@192.1.1.200 RST returned
Connect   443@192.1.1.200 RST returned

PC2> ping 192.1.1.100 -P 6 -p 443 -c 2

Connect   443@192.1.1.100 timeout
Connect   443@192.1.1.100 timeout

PC2> ping 192.1.1.200 -P 6 -p 80 -c 2

Connect   80@192.1.1.200 timeout
Connect   80@192.1.1.200 timeout
```

Figure 2.3: Rules of OUTSIDE to DMZ communication.

One thing to bear in mind is that we do not allow devices from the OUTSIDE network to start a communication with the INSIDE devices and the DMZ devices do not initiate communication neither with INSIDE nor OUTSIDE devices.



```
labcom@LabComServer:~$ ping 200.2.2.100
PING 200.2.2.100 (200.2.2.100) 56(84) bytes of data.
^C
--- 200.2.2.100 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 48ms

labcom@LabComServer:~$ ping 10.2.2.100
PING 10.2.2.100 (10.2.2.100) 56(84) bytes of data.
^C
--- 10.2.2.100 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 47ms
```

Figure 2.4: Rules of DMZ to INSIDE and OUTSIDE communication.

2.1 Policies and rules

Based on the services previously explained, we defined this zone policies and rules.

2.1.1 Zone INSIDE

In this zone, we implemented 2 policies: FROM-DMZ-TO-INSIDE and FROM-OUTSIDE-TO-INSIDE. Both of them just have 1 rule, which is only accepting established and related connections, preventing the DMZ and OUTSIDE zones to initiate any type of communication with the internal network. The configuration of this rules can be found in Section B.1, from line 1 to 4 and from line 49 to 52, respectively. This rules are assigned to a zone in lines 78 and 79.

2.1.2 Zone OUTSIDE

In this zone, we implemented 2 policies: FROM-DMZ-TO-OUTSIDE and FROM-INSIDE-TO-OUTSIDE. The former has 1 rule, similar to the rules of the INSIDE zone, which only accepts established and related connections, preventing the DMZ to initiate any communication to the Internet. The latter also has only 1 rule that accepts UDP traffic sent from the zone INSIDE, more specifically, from the internal network restricted to the range of source ports 5000 to 6000.

The configuration of this rules can be found in Section B.1, from line 6 to 9 and from line 33 to 36, respectively. This rules are assigned to a zone in lines 84 and 85.

2.1.3 Zone DMZ

In this zone, we implemented 2 policies: FROM-INSIDE-TO-DMZ and FROM-OUTSIDE-TO-DMZ. The latter has 2 rules, the first one accepts ICMP Echo Requests from the OUTSIDE zone to the public page hosted in the DMZ server (192.1.1.200/32), while the second one allows the OUTSIDE zone to access via HTTPS (TCP on port 443) to the ”public page” on the server. The former has 4 rules: the first one that accepts

ICMP Echo Requests from the INSIDE zone to all of DMZ zone, the second one that accepts HTTP, HTTPS and SSH from the Admin Network (10.10.10.0/24) to the DMZ, except for the 192.1.1.150, since it's a VPCS template used as a DNS Server, therefore doesn't support TCP communications so easily, the third rule that permits DNS (UDP using port 53) access on the DMZ from the zone INSIDE and finally, the forth rule that allows the other network in the internal network to access the DMZ zone with HTTP and HTTPS only (TCP using ports 80 and 443, respectively).

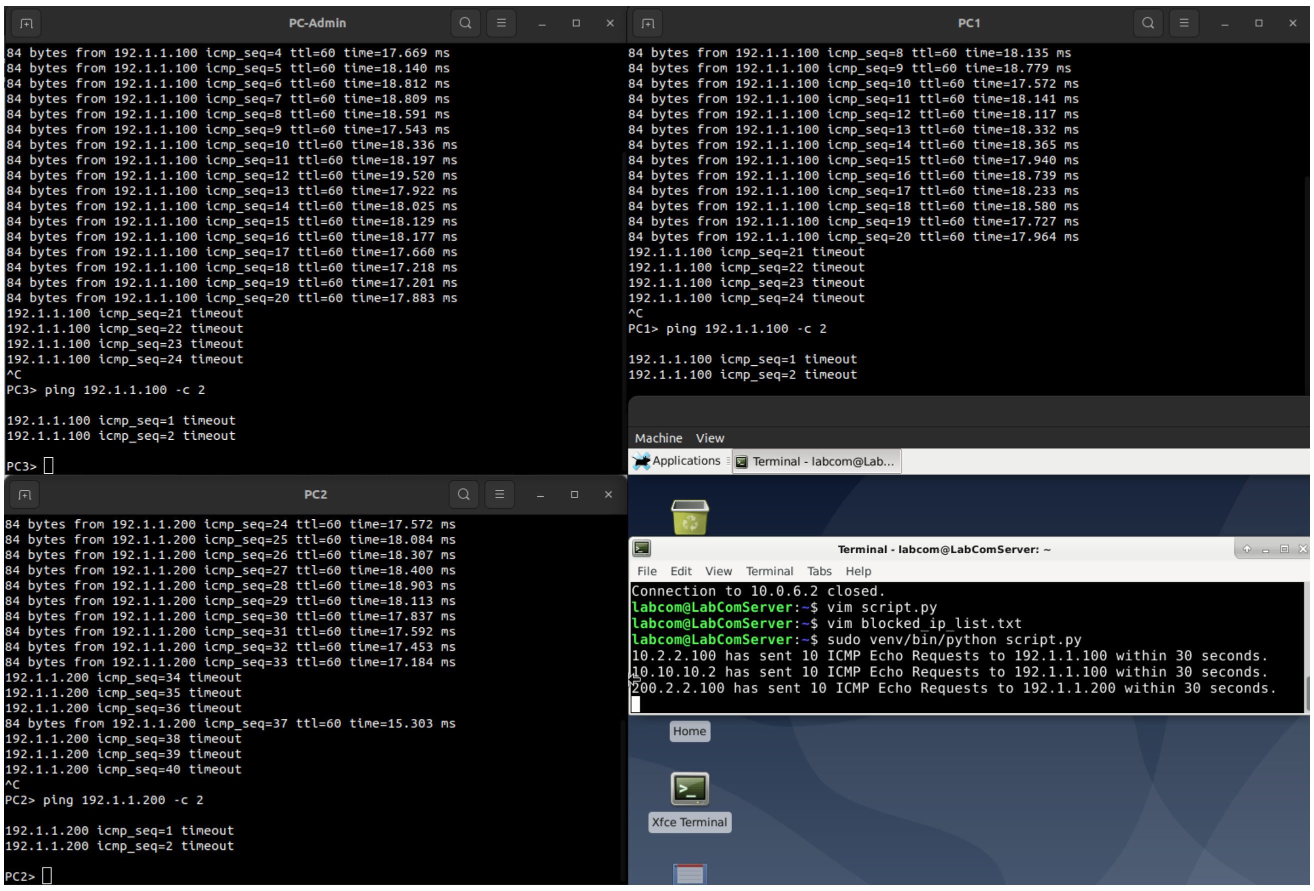
The configuration of this rules can be found in Section B.1, from line 11 to 31 and from line 38 to 47, respectively. This rules are assigned to a zone in lines 73 and 74.

2.2 DoS and DDoS attack denial

To simulate defense against an attack such as DoS or DDoS, we took the following steps. We created a Python script that runs on the DMZ-Server and detects IPs that send more than 10 pings within a 30-second period to the DMZ-Server IPs (192.1.1.100 or 192.1.1.200). Upon detecting such activity, the script adds the detected IP to a .txt file (blocked_ip_list.txt) and injects rules into the FWs to block these IPs.

First, we create the address groups for the detected IPs in the INSIDE zone (INSIDE_BLOCKED_IPS) and the OUTSIDE zone (OUTSIDE_BLOCKED_IPS), and add the respective IPs depending on the network (it is part of the INSIDE network if it belongs to 10.0.0.0/8, and OUTSIDE for all others). Next, we create a rule in the firewall named FROM-INSIDE-TO-DMZ that blocks all access to the DMZ for the address group INSIDE_BLOCKED_IPS. We perform a similar action for the firewall named FROM-OUTSIDE-TO-DMZ by blocking all access to the DMZ for the address group OUTSIDE_BLOCKED_IPS.

We demonstrate the behaviour of our monitoring script in Figure 2.5, despite our monitor being set to trigger after a computer sends 10 pings within 30 seconds, the lengthy duration of communication with the FW allows the PC to to do another 10 pings or even more. The time it takes is so slow that we even see on the PC2 that one of the firewalls takes much longer than the other, and therefore, one of the pings still gets through the firewall that is not yet changed. After that we see that all IPs are blocked because we used an address-group on blocking rule.



```
84 bytes from 192.1.1.100 icmp_seq=4 ttl=60 time=17.669 ms
84 bytes from 192.1.1.100 icmp_seq=5 ttl=60 time=18.140 ms
84 bytes from 192.1.1.100 icmp_seq=6 ttl=60 time=18.812 ms
84 bytes from 192.1.1.100 icmp_seq=7 ttl=60 time=18.809 ms
84 bytes from 192.1.1.100 icmp_seq=8 ttl=60 time=18.591 ms
84 bytes from 192.1.1.100 icmp_seq=9 ttl=60 time=17.543 ms
84 bytes from 192.1.1.100 icmp_seq=10 ttl=60 time=18.336 ms
84 bytes from 192.1.1.100 icmp_seq=11 ttl=60 time=18.197 ms
84 bytes from 192.1.1.100 icmp_seq=12 ttl=60 time=19.520 ms
84 bytes from 192.1.1.100 icmp_seq=13 ttl=60 time=17.922 ms
84 bytes from 192.1.1.100 icmp_seq=14 ttl=60 time=18.025 ms
84 bytes from 192.1.1.100 icmp_seq=15 ttl=60 time=18.129 ms
84 bytes from 192.1.1.100 icmp_seq=16 ttl=60 time=18.177 ms
84 bytes from 192.1.1.100 icmp_seq=17 ttl=60 time=17.660 ms
84 bytes from 192.1.1.100 icmp_seq=18 ttl=60 time=17.218 ms
84 bytes from 192.1.1.100 icmp_seq=19 ttl=60 time=17.201 ms
84 bytes from 192.1.1.100 icmp_seq=20 ttl=60 time=17.883 ms
192.1.1.100 icmp_seq=21 timeout
192.1.1.100 icmp_seq=22 timeout
192.1.1.100 icmp_seq=23 timeout
192.1.1.100 icmp_seq=24 timeout
^C
PC3> ping 192.1.1.100 -c 2

192.1.1.100 icmp_seq=1 timeout
192.1.1.100 icmp_seq=2 timeout

PC3> 
```

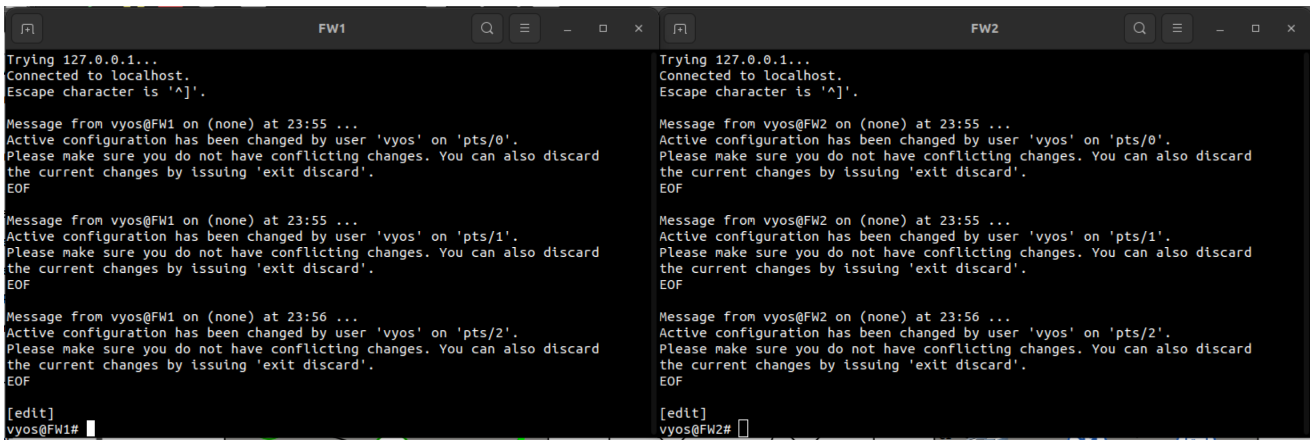
```
84 bytes from 192.1.1.100 icmp_seq=8 ttl=60 time=18.135 ms
84 bytes from 192.1.1.100 icmp_seq=9 ttl=60 time=18.779 ms
84 bytes from 192.1.1.100 icmp_seq=10 ttl=60 time=17.572 ms
84 bytes from 192.1.1.100 icmp_seq=11 ttl=60 time=18.141 ms
84 bytes from 192.1.1.100 icmp_seq=12 ttl=60 time=18.117 ms
84 bytes from 192.1.1.100 icmp_seq=13 ttl=60 time=18.332 ms
84 bytes from 192.1.1.100 icmp_seq=14 ttl=60 time=18.365 ms
84 bytes from 192.1.1.100 icmp_seq=15 ttl=60 time=17.940 ms
84 bytes from 192.1.1.100 icmp_seq=16 ttl=60 time=18.739 ms
84 bytes from 192.1.1.100 icmp_seq=17 ttl=60 time=18.233 ms
84 bytes from 192.1.1.100 icmp_seq=18 ttl=60 time=18.580 ms
84 bytes from 192.1.1.100 icmp_seq=19 ttl=60 time=17.727 ms
84 bytes from 192.1.1.100 icmp_seq=20 ttl=60 time=17.964 ms
192.1.1.100 icmp_seq=21 timeout
192.1.1.100 icmp_seq=22 timeout
192.1.1.100 icmp_seq=23 timeout
192.1.1.100 icmp_seq=24 timeout
^C
PC1> ping 192.1.1.100 -c 2

192.1.1.100 icmp_seq=1 timeout
192.1.1.100 icmp_seq=2 timeout
```

```
84 bytes from 192.1.1.200 icmp_seq=24 ttl=60 time=17.572 ms
84 bytes from 192.1.1.200 icmp_seq=25 ttl=60 time=18.084 ms
84 bytes from 192.1.1.200 icmp_seq=26 ttl=60 time=18.307 ms
84 bytes from 192.1.1.200 icmp_seq=27 ttl=60 time=18.400 ms
84 bytes from 192.1.1.200 icmp_seq=28 ttl=60 time=18.903 ms
84 bytes from 192.1.1.200 icmp_seq=29 ttl=60 time=18.113 ms
84 bytes from 192.1.1.200 icmp_seq=30 ttl=60 time=17.837 ms
84 bytes from 192.1.1.200 icmp_seq=31 ttl=60 time=17.592 ms
84 bytes from 192.1.1.200 icmp_seq=32 ttl=60 time=17.453 ms
84 bytes from 192.1.1.200 icmp_seq=33 ttl=60 time=17.184 ms
192.1.1.200 icmp_seq=34 timeout
192.1.1.200 icmp_seq=35 timeout
192.1.1.200 icmp_seq=36 timeout
84 bytes from 192.1.1.200 icmp_seq=37 ttl=60 time=15.303 ms
192.1.1.200 icmp_seq=38 timeout
192.1.1.200 icmp_seq=39 timeout
192.1.1.200 icmp_seq=40 timeout
^C
PC2> ping 192.1.1.200 -c 2

192.1.1.200 icmp_seq=1 timeout
192.1.1.200 icmp_seq=2 timeout

PC2> 
```

```
Connection to 10.0.6.2 closed.
labcom@LabComServer:~$ vim script.py
labcom@LabComServer:~$ vim blocked_ip_list.txt
labcom@LabComServer:~$ sudo venv/bin/python script.py
10.2.2.100 has sent 10 ICMP Echo Requests to 192.1.1.100 within 30 seconds.
10.10.10.2 has sent 10 ICMP Echo Requests to 192.1.1.100 within 30 seconds.
200.2.2.100 has sent 10 ICMP Echo Requests to 192.1.1.200 within 30 seconds.
```

Figure 2.5: Denial of Service Protection.

To further prove that our script is working wee see that firstly there are commited changes in our FWs after the script as been triggered as we can see on Figure 2.6, and on the Figure 2.7 we can see the before and after of our FWs configurations.



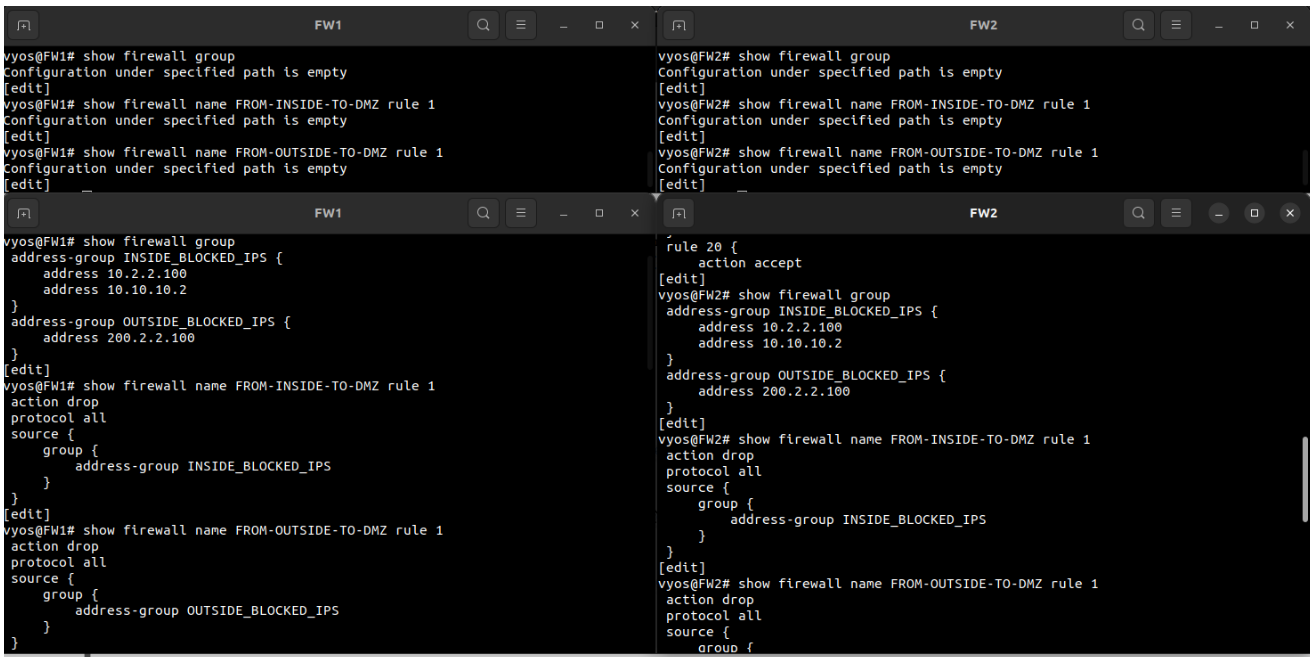
FW1

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Message from vyos@FW1 on (none) at 23:55 ...
Active configuration has been changed by user 'vyos' on 'pts/0'.
Please make sure you do not have conflicting changes. You can also discard
the current changes by issuing 'exit discard'.
EOF

Message from vyos@FW1 on (none) at 23:55 ...
Active configuration has been changed by user 'vyos' on 'pts/1'.
Please make sure you do not have conflicting changes. You can also discard
the current changes by issuing 'exit discard'.
EOF

Message from vyos@FW1 on (none) at 23:56 ...
Active configuration has been changed by user 'vyos' on 'pts/2'.
Please make sure you do not have conflicting changes. You can also discard
the current changes by issuing 'exit discard'.
EOF

[edit]
vyos@FW1#
```

FW2

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Message from vyos@FW2 on (none) at 23:55 ...
Active configuration has been changed by user 'vyos' on 'pts/0'.
Please make sure you do not have conflicting changes. You can also discard
the current changes by issuing 'exit discard'.
EOF

Message from vyos@FW2 on (none) at 23:55 ...
Active configuration has been changed by user 'vyos' on 'pts/1'.
Please make sure you do not have conflicting changes. You can also discard
the current changes by issuing 'exit discard'.
EOF

Message from vyos@FW2 on (none) at 23:56 ...
Active configuration has been changed by user 'vyos' on 'pts/2'.
Please make sure you do not have conflicting changes. You can also discard
the current changes by issuing 'exit discard'.
EOF

[edit]
vyos@FW2#
```

Figure 2.6: Commit being done remotely.



FW1

```
vyos@FW1# show firewall group
Configuration under specified path is empty
[edit]
vyos@FW1# show firewall name FROM-INSIDE-TO-DMZ rule 1
Configuration under specified path is empty
[edit]
vyos@FW1# show firewall name FROM-OUTSIDE-TO-DMZ rule 1
Configuration under specified path is empty
[edit]
```

FW2

```
vyos@FW2# show firewall group
Configuration under specified path is empty
[edit]
vyos@FW2# show firewall name FROM-INSIDE-TO-DMZ rule 1
Configuration under specified path is empty
[edit]
vyos@FW2# show firewall name FROM-OUTSIDE-TO-DMZ rule 1
Configuration under specified path is empty
[edit]
```

FW1

```
vyos@FW1# show firewall group
 address-group INSIDE_BLOCKED_IPS {
     address 10.2.2.100
     address 10.10.10.2
 }
 address-group OUTSIDE_BLOCKED_IPS {
     address 200.2.2.100
 }
[edit]
vyos@FW1# show firewall name FROM-INSIDE-TO-DMZ rule 1
 action drop
 protocol all
 source {
     group {
         address-group INSIDE_BLOCKED_IPS
     }
 }
[edit]
vyos@FW1# show firewall name FROM-OUTSIDE-TO-DMZ rule 1
 action drop
 protocol all
 source {
     group {
         address-group OUTSIDE_BLOCKED_IPS
     }
 }
```

FW2

```
 rule 20 {
     action accept
[edit]
vyos@FW2# show firewall group
 address-group INSIDE_BLOCKED_IPS {
     address 10.2.2.100
     address 10.10.10.2
 }
 address-group OUTSIDE_BLOCKED_IPS {
     address 200.2.2.100
 }
[edit]
vyos@FW2# show firewall name FROM-INSIDE-TO-DMZ rule 1
 action drop
 protocol all
 source {
     group {
         address-group INSIDE_BLOCKED_IPS
     }
 }
[edit]
vyos@FW2# show firewall name FROM-OUTSIDE-TO-DMZ rule 1
 action drop
 protocol all
 source {
     group {
```

Figure 2.7: Firewalls (1 and 2) configuration before (upper) and after (lower) DDoS.

There are two considerations to keep in mind. First, the most suitable monitoring devices shouldn't be the DMZ. In an ideal scenario, this device should have access to the networks connecting the FWs to the DMZ and have a direct connection to the FWs. Its sole function would be to monitor the network with the use of the script and should only be accessible by terminals within the Admin network or, in an extreme scenario, by a specific address within the Admin network.

Another factor to consider is that if we were to change the monitoring device for detecting attacks, we would need to update the listen-address accordingly to match the new location of the monitoring device.

On a side note, it's also relevant to inform that the property of being able to block internal IPs is available for the worst-case scenario where a machine of the internal network may become compromised.

It is essential to mention that despite having prevention measures for DDoS and DoS attacks in place, the synchronization between LBs could potentially be exploited to launch an attack on the LBs themselves. The fact that the state is synchronized with conntrack-sync implies the exchange of information between LBs about the received traffic. A DDoS/DoS attack could potentially overwhelm the LBs by generating a large volume of traffic that they would need to process and synchronize, which could lead to performance issues or even a disruption of service. One solution would be to avoid synchronizing the LBs. However, to still have load-balancing, we could employ a load-balancing algorithm such as iphash. This algorithm enables the consistent distribution of traffic to the appropriate interface based on the IP address, ensuring that traffic from a specific client is always directed to the same interface, thus maintaining session persistence without the need for synchronization.

In order to do this, we developed the python script present on Section C.

Appendix A

Load Balancers Configuration

A.1 Load Balancers 1

A.1.1 Load Balancer 1 A (LB1A)

```
set high-availability vrrp group LB1Cluster interface eth1
set high-availability vrrp group LB1Cluster rfc3768-compatibility
set high-availability vrrp group LB1Cluster virtual-address 10.0.0.1/24
set high-availability vrrp group LB1Cluster vrid 1
set high-availability vrrp sync-group LB1Cluster member LB1Cluster

set load-balancing wan disable-source-nat
set load-balancing wan interface-health eth2 nexthop 10.2.0.2
set load-balancing wan interface-health eth3 nexthop 10.2.1.2
set load-balancing wan rule 1 inbound-interface eth0
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan rule 1 interface eth3 weight 1
set load-balancing wan rule 1 protocol all
set load-balancing wan sticky-connections inbound

set protocols static route 10.2.2.0/24 next-hop 10.1.1.10
set protocols static route 10.10.10.0/24 next-hop 10.1.1.10

set service conntrack-sync accept-protocol tcp,udp,icmp
set service conntrack-sync disable-external-cache
set service conntrack-sync failover-mechanism vrrp sync-group LB1Cluster
set service conntrack-sync interface eth1
set service conntrack-sync mcast-group 225
```

A.1.2 Load Balancer 1 B (LB1B)

```
set high-availability vrrp group LB1Cluster interface eth1
set high-availability vrrp group LB1Cluster rfc3768-compatibility
set high-availability vrrp group LB1Cluster virtual-address 10.0.0.1/24
set high-availability vrrp group LB1Cluster vrid 1
set high-availability vrrp sync-group LB1Cluster member LB1Cluster

set load-balancing wan disable-source-nat
set load-balancing wan interface-health eth2 nexthop 10.2.3.1
set load-balancing wan interface-health eth3 nexthop 10.2.4.1
set load-balancing wan rule 1 inbound-interface eth0
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan rule 1 interface eth3 weight 1
set load-balancing wan rule 1 protocol all
set load-balancing wan sticky-connections inbound

set protocols static route 10.2.2.0/24 next-hop 10.1.1.10
set protocols static route 10.10.10.0/24 next-hop 10.1.1.10

set service conntrack-sync accept-protocol tcp,udp,icmp
set service conntrack-sync disable-external-cache
set service conntrack-sync failover-mechanism vrrp sync-group LB1Cluster
set service conntrack-sync interface eth1
set service conntrack-sync mcast-group 225
```

A.2 Load Balancers 2

A.2.1 Load Balancer 2 A (LB2A)

```
set high-availability vrrp group LB2Cluster interface eth1
set high-availability vrrp group LB2Cluster rfc3768-compatibility
set high-availability vrrp group LB2Cluster virtual-address ...
    192.168.100.1/24
set high-availability vrrp group LB2Cluster vrid 2
set high-availability vrrp sync-group LB2Cluster member LB2Cluster

set load-balancing wan disable-source-nat
set load-balancing wan interface-health eth2 nexthop 10.0.3.2
set load-balancing wan interface-health eth3 nexthop 10.0.1.2
set load-balancing wan rule 1 inbound-interface eth0
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan rule 1 interface eth3 weight 1
set load-balancing wan rule 1 protocol all
set load-balancing wan sticky-connections inbound

set protocols static route 200.2.2.0/24 next-hop 200.1.1.10

set service conntrack-sync accept-protocol tcp,udp,icmp
set service conntrack-sync disable-external-cache
set service conntrack-sync failover-mechanism vrrp sync-group LB2Cluster
set service conntrack-sync interface eth1
set service conntrack-sync mcast-group 225.0.0.50
```

A.2.2 Load Balancer 2 B (LB2B)

```
set high-availability vrrp group LB2Cluster interface eth1
set high-availability vrrp group LB2Cluster rfc3768-compatibility
set high-availability vrrp group LB2Cluster virtual-address ...
    192.168.100.1/24
set high-availability vrrp group LB2Cluster vrid 2
set high-availability vrrp sync-group LB2Cluster member LB2Cluster

set load-balancing wan disable-source-nat
set load-balancing wan interface-health eth2 nexthop 10.0.4.1
set load-balancing wan interface-health eth3 nexthop 10.0.2.1
set load-balancing wan rule 1 inbound-interface eth0
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan rule 1 interface eth3 weight 1
set load-balancing wan rule 1 protocol all
set load-balancing wan sticky-connections inbound

set protocols static route 200.2.2.0/24 next-hop 200.1.1.10

set service conntrack-sync accept-protocol tcp,udp,icmp
set service conntrack-sync disable-external-cache
set service conntrack-sync failover-mechanism vrrp sync-group LB2Cluster
set service conntrack-sync interface eth1
set service conntrack-sync mcast-group 225.0.0.50
```

A.3 Load Balancer DMZ

```
set load-balancing wan disable-source-nat
set load-balancing wan interface-health eth1 nexthop 10.0.6.2
set load-balancing wan interface-health eth2 nexthop 10.0.7.2
set load-balancing wan rule 1 inbound-interface eth0
set load-balancing wan rule 1 interface eth1 weight 1
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan rule 1 protocol all
set load-balancing wan sticky-connections inbound
```

Appendix B

Firewalls Configuration

B.1 Firewall 1

```
set firewall name FROM-DMZ-TO-INSIDE rule 10 action accept
set firewall name FROM-DMZ-TO-INSIDE rule 10 description "Accept ...
    Established-related connections"
set firewall name FROM-DMZ-TO-INSIDE rule 10 state established enable
set firewall name FROM-DMZ-TO-INSIDE rule 10 state related enable

set firewall name FROM-DMZ-TO-OUTSIDE rule 10 action accept
set firewall name FROM-DMZ-TO-OUTSIDE rule 10 description "Accept ...
    Established-related connections"
set firewall name FROM-DMZ-TO-OUTSIDE rule 10 state established enable
set firewall name FROM-DMZ-TO-OUTSIDE rule 10 state related enable

set firewall name FROM-INSIDE-TO-DMZ rule 10 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 10 description "Accept ICMP ...
    Echo Request to DMZ"
set firewall name FROM-INSIDE-TO-DMZ rule 10 destination address ...
    192.1.1.0/24
set firewall name FROM-INSIDE-TO-DMZ rule 10 icmp type 8
set firewall name FROM-INSIDE-TO-DMZ rule 10 protocol icmp
set firewall name FROM-INSIDE-TO-DMZ rule 20 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 20 description "Accept ...
    HTTP, HTTPS and SSH from Admin to DMZ"
set firewall name FROM-INSIDE-TO-DMZ rule 20 destination address ...
    192.1.1.0/24
set firewall name FROM-INSIDE-TO-DMZ rule 20 destination port 22,80,443
set firewall name FROM-INSIDE-TO-DMZ rule 20 protocol tcp
set firewall name FROM-INSIDE-TO-DMZ rule 20 source address 10.10.10.0/24
set firewall name FROM-INSIDE-TO-DMZ rule 30 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 30 description "Allow DNS ...
    access to DMZ"
set firewall name FROM-INSIDE-TO-DMZ rule 30 destination address ...
    192.1.1.0/24
set firewall name FROM-INSIDE-TO-DMZ rule 30 destination port 53
set firewall name FROM-INSIDE-TO-DMZ rule 30 protocol udp
set firewall name FROM-INSIDE-TO-DMZ rule 40 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 40 description "Accept HTTP ...
    and HTTPS from INSIDE to DMZ"
set firewall name FROM-INSIDE-TO-DMZ rule 40 destination address ...
    192.1.1.0/24
set firewall name FROM-INSIDE-TO-DMZ rule 40 destination port 80,443
set firewall name FROM-INSIDE-TO-DMZ rule 40 protocol tcp

set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 action accept
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 description "Accept ...
    UDP from ports 5000-6000"
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 destination port ...
    5000-6000
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 protocol udp

set firewall name FROM-OUTSIDE-TO-DMZ rule 10 action accept
set firewall name FROM-OUTSIDE-TO-DMZ rule 10 description "Accept ...
    ICMP Echo Request to Public Page"
set firewall name FROM-OUTSIDE-TO-DMZ rule 10 destination address ...
    192.1.1.200/32
set firewall name FROM-OUTSIDE-TO-DMZ rule 10 icmp type 8
set firewall name FROM-OUTSIDE-TO-DMZ rule 10 protocol icmp
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 action accept
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 description "Accept ...
    HTTPS to Public Page"
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 destination address ...
    192.1.1.200/32
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 destination port 443
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 protocol tcp

set firewall name FROM-OUTSIDE-TO-INSIDE rule 10 action accept
set firewall name FROM-OUTSIDE-TO-INSIDE rule 10 description "Accept ...
    Established-related connections"
set firewall name FROM-OUTSIDE-TO-INSIDE rule 10 state established enable
set firewall name FROM-OUTSIDE-TO-INSIDE rule 10 state related enable

set nat source rule 10 outbound-interface eth0
set nat source rule 10 source address 10.0.0.0/8
set nat source rule 10 translation address 192.1.0.1-192.1.0.10

set nat source rule 20 outbound-interface eth3
set nat source rule 20 source address 10.0.0.0/8
set nat source rule 20 translation address 192.1.0.1-192.1.0.10

set protocols static route 0.0.0.0/0 next-hop 10.0.1.1
set protocols static route 0.0.0.0/0 next-hop 10.0.2.2
set protocols static route 10.0.0.0/12 next-hop 10.2.1.1
set protocols static route 10.0.0.0/12 next-hop 10.2.4.2
set protocols static route 192.1.1.0/24 next-hop 10.0.6.1

set service ssh access-control allow user vyos
set service ssh listen-address 10.0.6.2
set service ssh port 22

set zone-policy zone DMZ description DMZ
set zone-policy zone DMZ from INSIDE firewall name FROM-INSIDE-TO-DMZ
set zone-policy zone DMZ from OUTSIDE firewall name FROM-OUTSIDE-TO-DMZ
set zone-policy zone DMZ interface eth4

set zone-policy zone INSIDE description "Inside (Internal Network)"
set zone-policy zone INSIDE from DMZ firewall name FROM-DMZ-TO-INSIDE
set zone-policy zone INSIDE from OUTSIDE firewall name ...
    FROM-OUTSIDE-TO-INSIDE
set zone-policy zone INSIDE interface eth1
set zone-policy zone INSIDE interface eth2

set zone-policy zone OUTSIDE description "Outside (Internet)"
set zone-policy zone OUTSIDE from DMZ firewall name FROM-DMZ-TO-OUTSIDE
set zone-policy zone OUTSIDE from INSIDE firewall name ...
    FROM-INSIDE-TO-OUTSIDE
set zone-policy zone OUTSIDE interface eth0
set zone-policy zone OUTSIDE interface eth3
```

B.2 Firewall 2

```
-- the same firewall rules defined in Firewall 1

set nat source rule 10 outbound-interface eth0
set nat source rule 10 source address 10.0.0.0/8
set nat source rule 10 translation address 192.1.0.11-192.1.0.20

set nat source rule 20 outbound-interface eth3
set nat source rule 20 source address 10.0.0.0/8
set nat source rule 20 translation address 192.1.0.11-192.1.0.20

set protocols static route 0.0.0.0/0 next-hop 10.0.3.1
set protocols static route 0.0.0.0/0 next-hop 10.0.4.2
set protocols static route 10.0.0.0/12 next-hop 10.2.0.1
set protocols static route 10.0.0.0/12 next-hop 10.2.3.2
set protocols static route 192.1.1.0/24 next-hop 10.0.7.1

set service ssh access-control allow user vyos
set service ssh listen-address 10.0.7.2
set service ssh port 22

-- the same zones defined in Firewall 1
```

Appendix C

Script for attack detection

```
from scapy.all import *
from collections import defaultdict
from threading import Lock, Timer
from netmiko import ConnectHandler
import time

TARGET_IPS = ['192.1.1.100' , '192.1.1.200']
THRESHOLD = 10
TIME_WINDOW = 30
BLOCKED_IP_FILE = "blocked_ip_list.txt"

icmp_counts = defaultdict (lambda: defaultdict (int))
icmp_times = defaultdict(lambda: defaultdict (list))
detected_ips = set()
lock = Lock()

def block_ip_on_vyos(ip):
    fw = {
        "device_type": "vyos",
        "host": ip,
        "username": "vyos",
        "password": "vyos",
        "port": 22
    }

    net_connect = ConnectHandler(**fw)

    inside_ips = []
    outside_ips = []

    with open(BLOCKED_IP_FILE, 'r') as file:
        for line in file:
            ip = line.strip()
            if ip[0:3] == "10.":
                inside_ips.append(ip)
            else:
                outside_ips.append(ip)

    if inside_ips != []:
        config_commands = ['del firewall group address-group ...
   INSIDE_BLOCKED_IPS']
        for ip in inside_ips:
            config_commands.append(f'set firewall group address-...
   group INSIDE_BLOCKED_IPS address {ip}')
        config_commands.append('set firewall name FROM-INSIDE-TO-DMZ...
    rule 1 action drop')
        config_commands.append('set firewall name FROM-INSIDE-TO-DMZ...
    rule 1 protocol all')
        config_commands.append('set firewall name FROM-INSIDE-TO-DMZ...
    rule 1 source group address-group INSIDE_BLOCKED_IPS')
        output = net_connect.send_config_set(config_commands, ...
   exit_config_mode=False)
        print (output)
        output = net_connect.commit()
        print(output)
    if outside_ips != []:
        config_commands = ['del firewall group address-group ...
   OUTSIDE_BLOCKED_IPS']
        for ip in outside_ips:
            config_commands.append(f'set firewall group address-...
   group OUTSIDE_BLOCKED_IPS address {ip}')
```

```
        config_commands.commands('set firewall name FROM-OUTSIDE-TO-...
    DMZ rule 1 action drop')
        config_commands.commands('set firewall name FROM-OUTSIDE-TO-...
    DMZ rule 1 protocol all')
        config_commands.commands('set firewall name FROM-OUTSIDE-TO-...
    DMZ rule 1 source group address-group OUTSIDE_BLOCKED_IPS')
        output = net_connect.send_config_set(config_commands, ...
    exit_config_mode=False)
        print (output)
        output = net_connect.commit()
        print(output)

def reset_counts():
    global icmp_counts
    global icmp_times
    with lock:
        icmp_counts = defaultdict (lambda: defaultdict (int))
        icmp_times = defaultdict (lambda: defaultdict (list))
    Timer(TIME_WINDOW, reset_counts).start()

def update_blocked_ip_file(ip):
    if not string_in_file(BLOCKED_IP_FILE, ip):
        with open (BLOCKED_IP_FILE, "a") as f:
            f.write(f"{ip}\n")

def string_in_file(filename, target_string):
    with open(filename, 'r') as file:
        for line in file:
            if target_string in line:
                return False
    return True

def monitor_icmp(packet):
    if packet.haslayer(ICMP) and packet[ICMP].type == 8:    # ICMP ...
    Echo Request
        src_ip = packet[IP].src
        dst_ip = packet[IP].dst

    if dst_ip in TARGET_IPS:
        with lock:
            icmp_counts [src_ip][dst_ip] += 1
            icmp_times[src_ip][dst_ip].append(time.time())
            icmp_times [src_ip][dst_ip] = [ t for t in icmp_times [...
    src_ip][dst_ip] if t > (time.time() - TIME_WINDOW) ]

            if icmp_counts [src_ip][dst_ip] ≥ THRESHOLD:
                if src_ip not in detected_ips:
                    detected_ips.add(src_ip)
                    update_blocked_ip_file(src_ip)
                    print (f"{src_ip} has sent {icmp_counts[src_ip][...
    dst_ip]} ICMP Echo Requests to {dst_ip} within {TIME_WINDOW} ...
    seconds.")
                    block_ip_vyos1_thread = threading.Thread(target=...
    block_ip_on_vyos, args=("10.0.6.2",))
                    block_ip_vyos2_thread = threading.Thread(target=...
    block_ip_on_vyos, args=("10.0.7.2",))
                    block_ip_vyos1_thread.start()
                    block_ip_vyos2_thread.start()
                    block_ip_vyos1_thread.join()
                    block_ip_vyos2_thread.join()
                icmp_counts[src_ip][dst_ip] = 0
                icmp_times[src_ip][dst_ip] = []

reset_counts ()
sniff(prn=monitor_icmp)
```