Universidade de Aveiro

Segurança em Redes de Comunicações

Tiago Marques 98459, Bruno Lemos 98221

Contents

Chapter 1

The Network

1.1 Initial Configurations

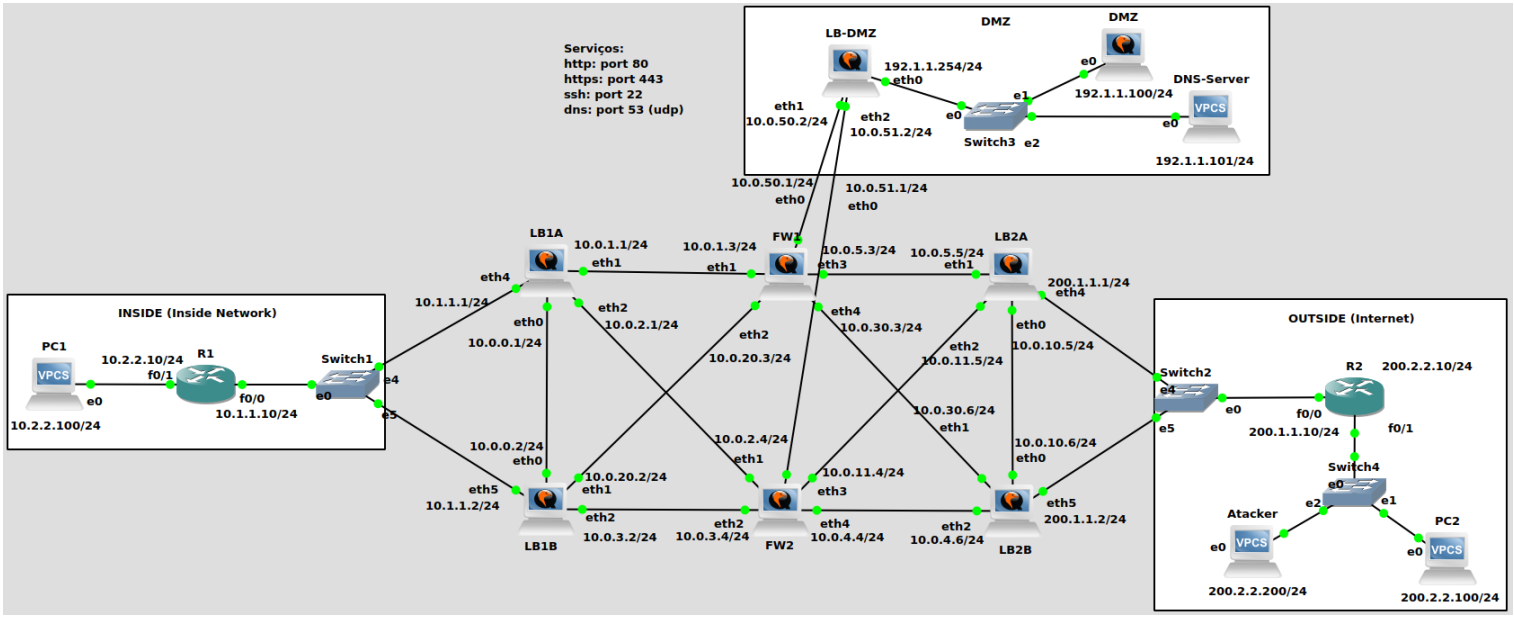

Figure 1.1: Network Layout

The devices interfaces IPs are configured as shown in the figure above. Since there is no active routing protocol, we have static routes in order to route traffic, they are present in:

- R1, to forward any traffic that does not match any of specific routes to the LB1A and LB1B interfaces that belong to the 10.1.1.0/24 network.
- R2, to forward traffic destined to the IP range 192.1.0.0/23 to the LB2A and LB2B interfaces that belong to the 200.1.1.0/24 network.
- LB1(A/B), to forward traffic destined to the IP range 10.2.2.0/24 to the R1 router interface with IP 10.1.1.10.
- LB2(A/B), to forward traffic destined to the IP range 200.2.2.0/24 to the R2 router interface with IP 200.1.1.10.
- FW(1/2), the traffic destined to the IP range 10.2.2.0/24 goes to one of the directly connected interfaces of the LB1(A/B) with the firewalls and any traffic that does not match any specific routes goes to one of the directly connected interfaces of the LB2(A/B) with the firewalls.

We still have a NAT mechanism implemented, in FW1 and FW2, in order to hide the internal topology of the network. The NAT pool consists of the 192.1.0.0/24 network divided into 4 smaller pools each pool for one of the eth3 and eth4 interfaces of FW1 and FW2.

1.2 Load Balancing with Redundancy and State Synchronization

The network makes use of 5 VyOS (LB1A, LB1B, LB2A, LB2B and LB-DMZ) to carry out the work of Load Balancers. When configuring this functionality, the VyOS, by assigning equal weights to the eth1 and eth2 interfaces, distributes the traffic coming from its inbound-interface with equal chance between the 2 firewalls (FW1 and FW2).

By also activating the sticky connections functionality, the load balancer tracks the state of each connection and ensures that subsequent packets for that connection are always routed through the same firewall. This means that the firewalls do not need to synchronize connection state information between each other, as the load balancer is responsible for maintaining the state information of each connection.

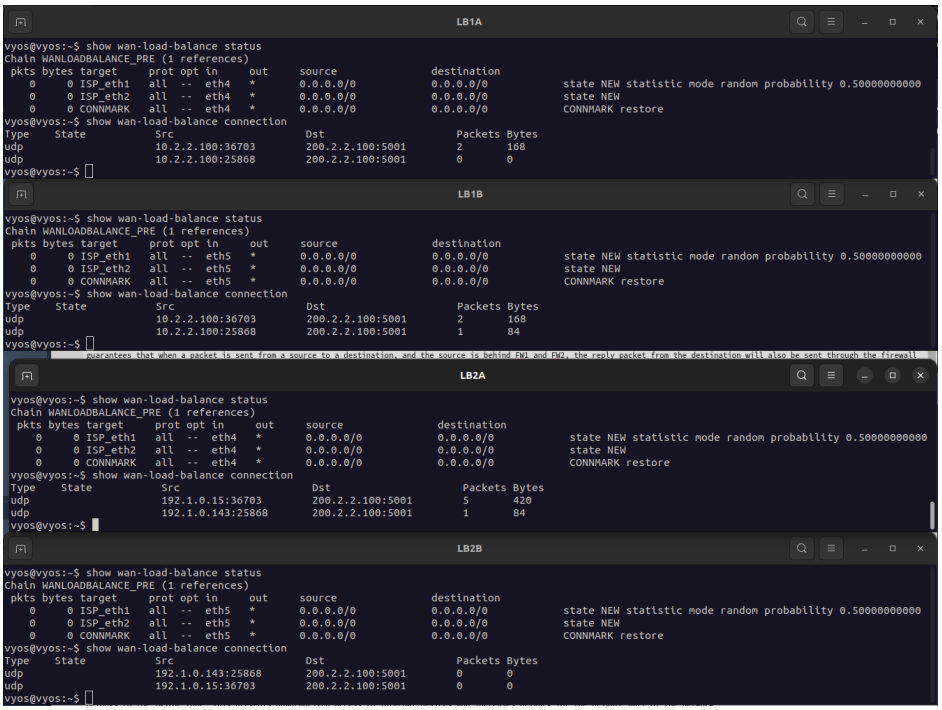

Figure 1.2: Load Balancing status and current connections of the LBs during a ping from PC1 to PC2

The network still makes use of 4 of those 5 Load Balancer to achieve redundancy, making 2 pairs of Load Balancers, (LB1:[LB1A and LB1B] and LB2:[LB2A and LB2B]). To get this scenario working we made use of conntrack-sync which allows a Load Balancer to track a connection and share that information with the other member of the pair, so that the connection information for each flow is synchronized between the pair in real-time, ensuring that the devices have the same state for each connection and thus ensuring redundancy.

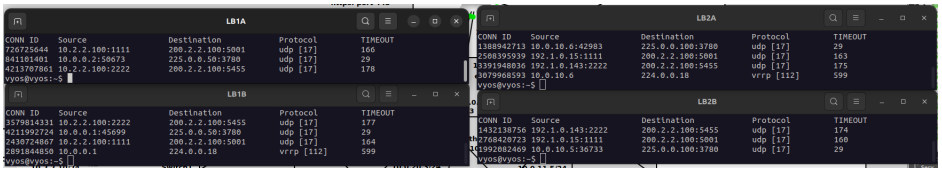

Figure 1.3: Current connection tracking for IPv4 synchronized for LB1A and LB1B and for LB2A and LB2B

As we can see in the screenshot above when executing the command 'show conntrack table ipv4' we notice that the pair LB1A and LB1B and the pair LB1A and LB1B have their current connections tracked and synced.

Now consider the pair of Load Balancers LB1, because the members trade flows with each other in real-time, during a DDoS attack the pair can become overwhelmed while trying to synchronize all the incoming flows which may cause the devices to fail. In order to reduce vulnerability to this type of attack, one solution would be to discard synchronization between them and use a load balance algorithm called IP Hash. This algorithm makes use of the client's IP and/or other attributes to determine which device will receive the flow, so although there is no longer synchronization, through the same hash function both members make take the same actions for incoming flows from the same client, ensuring consistency in the load balancing.

1.3 Policies Definition

As we can see, from the picture above, the network has 3 zones:

- INSIDE, a protected zone for the internal devices
- OUTSIDE, this zone is the internet
- DMZ, a semi-protected zone where the services/servers are exposed

Having these zones defined we implemented some policies to restrict the communication between them so the network can be safer, in order to do that we make use of 2 VyOS acting as firewalls (FW1 and FW2 in the figure 1.1).

Initially we restricted the access that the devices in the INSIDE and OUTSIDE zones had with each other, through the firewall FROM-INSIDE-TO-OUTSIDE, allowing traffic to pass from the INSIDE zone to the OUTSIDE zone only if:

1. Communication between zones is done using the UDP protocol.
2. Communication between zones is done with the destination port between 5000 and 6000

Additionally, we created another firewall, FROM-OUTSIDE-TO-INSIDE, which allows communication from OUTSIDE (PC2) to INSIDE (PC1) only if the device who initiated the communication belongs to the INSIDE zone. This prevents unauthorized access to internal devices and increases privacy for the private part of the network.

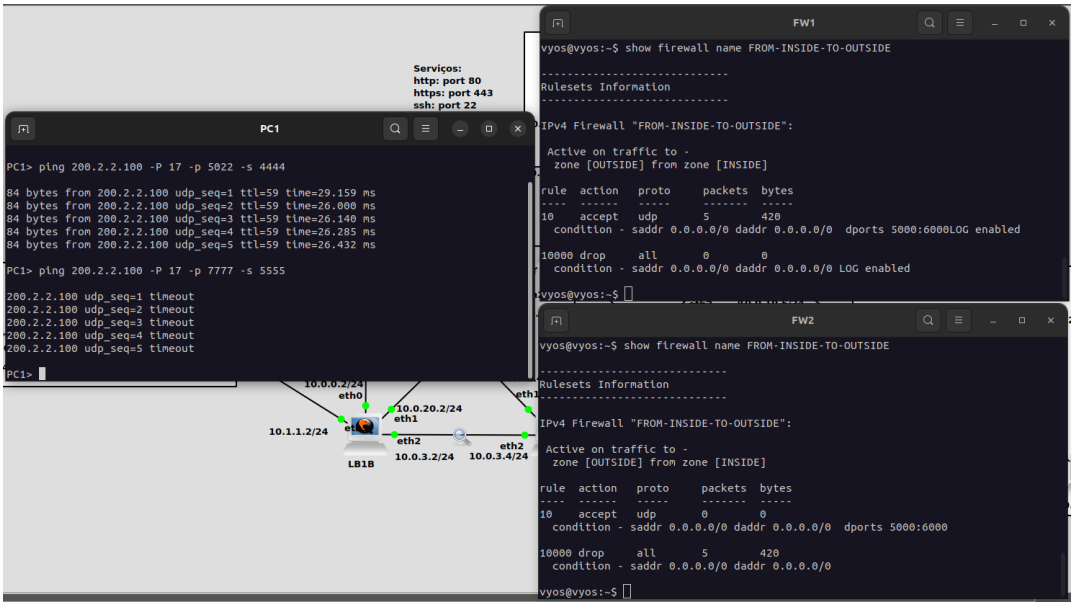

Figure 1.4: UDP Pings from PC1 to PC2

As we can see in the image above, PC1 has connectivity with PC2 if it follows the requirements listed above. The first ping *ping 200.2.2.100 -P 17 -p 5022 -s 4444* goes trough FW1 and is allowed to pass, while the second ping *ping 200.2.2.100 -P 17 -p 7777 -s 5555* goes trough FW2 and as it does not meet the requirements the packets are dropped.

The DMZ is a zone that has the following services:

- Http Service on 192.1.1.100:80
- Https Service on 192.1.1.100:443
- SSH Service on 192.1.1.100:22
- DNS Service on 192.1.1.101:53

As such, it is necessary to restrict which devices can access which services. We chose to allow devices in the INSIDE zone to access every available service. Firstly, we defined rules on the FROM-INSIDE-TO-DMZ firewall, as the name suggests it handles the packets originating from the INSIDE zone to the DMZ zone, the rules allow traffic if:

- The destination port must be 22 (SSH), 80 (Http) or 443 (Https), the destination address must be 192.1.1.100 (DMZ) and the used protocol must be TCP.
- For the DNS Service the destination port must be 53, the address 192.1.1.101 (DNS-Server) and the UDP protocol must be used.

The screenshot below demonstrates the connectivity of the INSIDE zone (PC1) with some of the services available in the DMZ

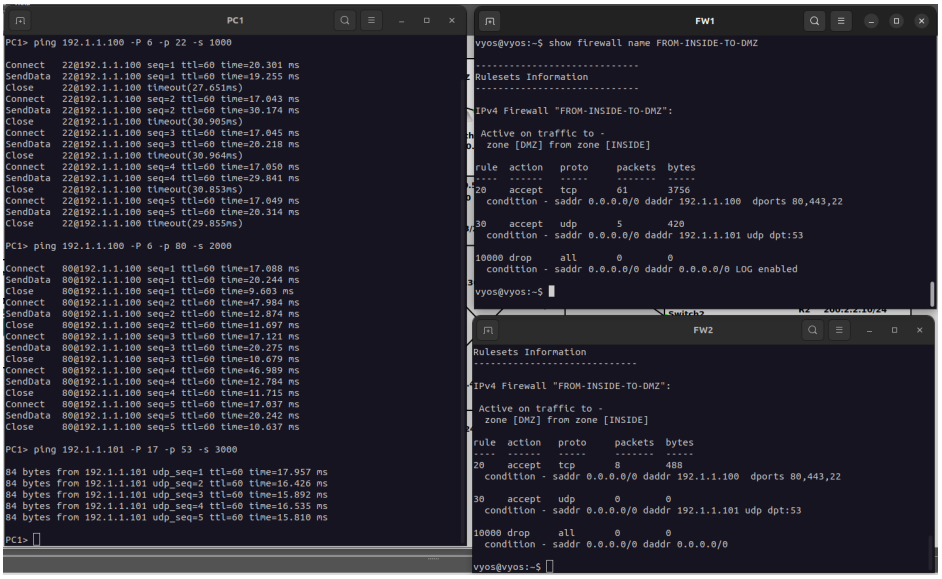

Figure 1.5: Pings from PC1 to test connectivity to Services

Now for internet devices trying to access our services in the DMZ zone, they can access all available services except the SSH service. Therefore, the rules defined in the FROM-OUTSIDE-TO-DMZ firewall are the same defined in the FROM-INSIDE-TO-DMZ firewall, with the exception of accessing port 22 for the SSH service.

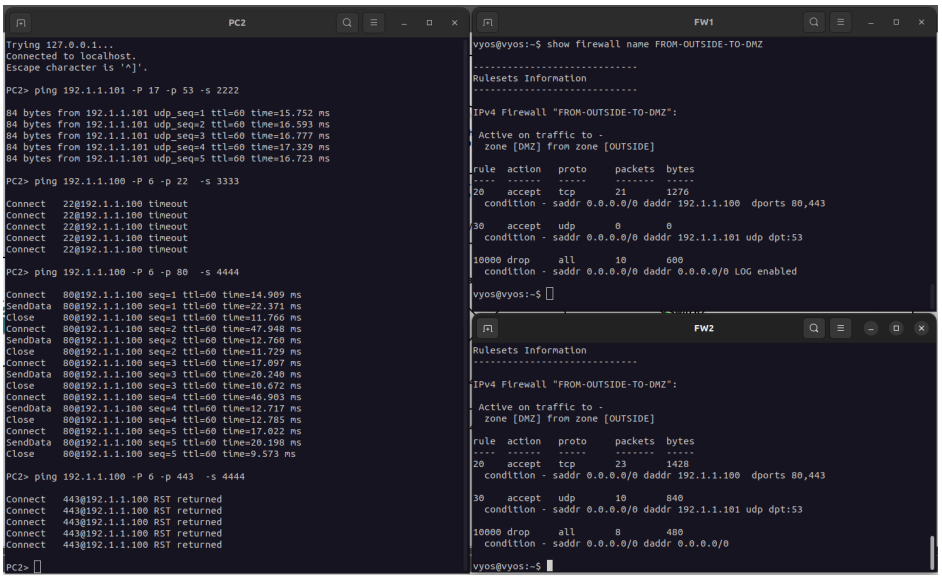

Figure 1.6: Pings from PC2 to test connectivity to Services

As we can check in the image above, the defined rules above work, the OUTSIDE zone (PC2) has connectivity to all services except for the SSH service, *ping 192.1.1.100 -P 6 -p 22 -s 3333*.

Finally, as the DMZ is an area exposed to the internet, we prevent it from starting communication on its own, thus reducing the risk of unauthorized access from the DMZ zone to other zones. With this in mind, we created 2 new firewalls, FROM-DMZ-TO-INSIDE and FROM-DMZ-TO-OUTSIDE, which only allow traffic to pass if it is in response to a request made to a device in the DMZ zone.

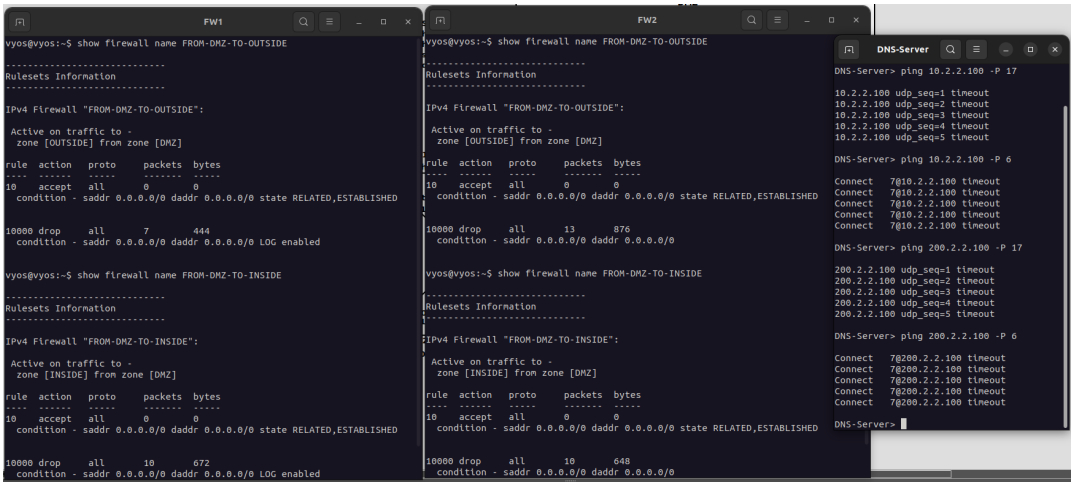

Figure 1.7: Device from DMZ zone tries to initiate communication with other zones unsuccessfully

1.4 Extra: Script to Block Suspect IP

We have developed a script that, when executed, blocks suspicious IPs, belonging to the internet, from communicating with the INSIDE and DMZ zones. This script connects via ssh to FW1 and FW2 and creates rules in which if the source address is the suspect IP the packets are dropped, these rules are then applied to the firewalls FROM-OUTSIDE-TO-INSIDE and FROM-OUTSIDE-TO-DMZ.

To test this scenario we have an 'Attacker' in the OUTSIDE zone, as shown in figure 1.1, this attacker is initially able to make requests to the DMZ zone but after running the script the 'Attacker' should no longer be able to communicate with any of the zones.

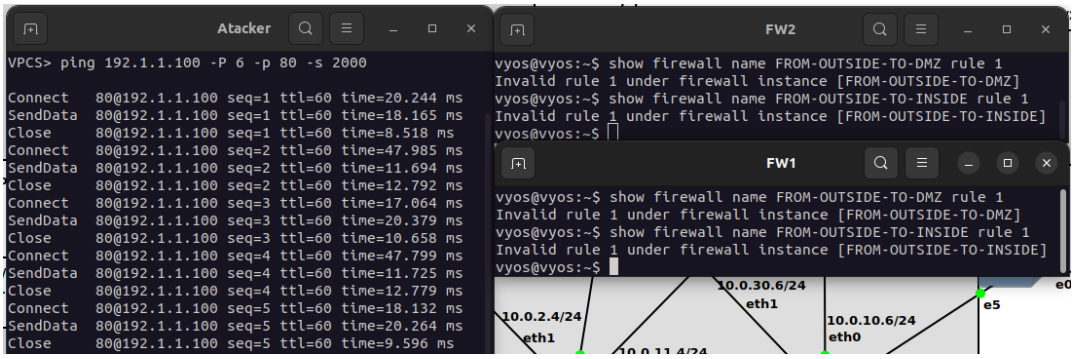

Figure 1.8: Attacker can access the services available on the DMZ

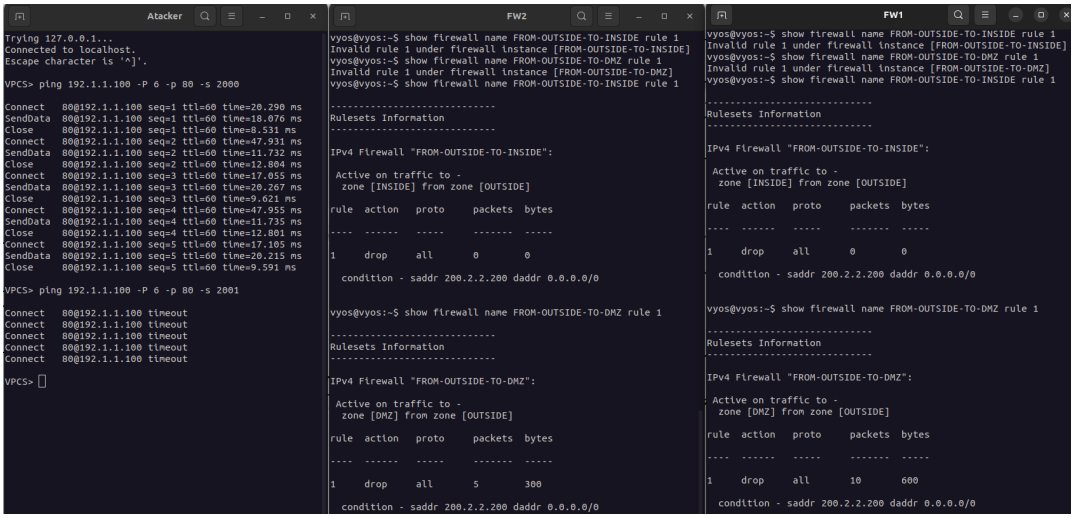

Figure 1.9: After the script is executed, the Attacker is denied access

As we can see, in figure 1.8, the Attacker is able to access the services available in the DMZ zone, but after running the script, figure 1.9, rule 1 is created and applied in order to block the Attacker's IP access to the zone INSIDE and DMZ.

Currently the script runs on the DMZ machine but ideally a new machine should be added and linked directly to FW1 and FW2. Since this machine can change firewall settings, access to it must be extremely restricted, allowing only a small range of internal devices to communicate with it, through new firewall rules and/or zones. Another possibility would be to ensure that this new machine can only communicate with the FW1 and FW2 devices to execute the script and nothing else.

.1 Appendix

.1.1 Static Routes

R1:
```
ip route 0.0.0.0 0.0.0.0 10.1.1.1
ip route 0.0.0.0 0.0.0.0 10.1.1.2
```

R2:
```
ip route 192.1.0.0 255.255.254.0 200.1.1.1
ip route 192.1.0.0 255.255.254.0 200.1.1.2
```

LB1A: `set protocols static route 10.2.2.0/24 next-hop 10.1.1.10`

LB1B: `set protocols static route 10.2.2.0/24 next-hop 10.1.1.10`

LB2A: `set protocols static route 200.2.2.0/24 next-hop 200.1.1.10`

LB2B: `set protocols static route 200.2.2.0/24 next-hop 200.1.1.10`

FW1:
```
set protocols static route 0.0.0.0/0 next-hop 10.0.5.5
set protocols static route 0.0.0.0/0 next-hop 10.0.30.6
set protocols static route 10.2.2.0/24 next-hop 10.0.1.1
set protocols static route 10.2.2.0/24 next-hop 10.0.20.2
set protocols static route 192.1.1.0/24 next-hop 10.0.50.2
```

FW2:
```
set protocols static route 0.0.0.0/0 next-hop 10.0.4.6
set protocols static route 0.0.0.0/0 next-hop 10.0.11.5
set protocols static route 10.2.2.0/24 next-hop 10.0.3.2
set protocols static route 10.2.2.0/24 next-hop 10.0.2.1
set protocols static route 192.1.1.0/24 next-hop 10.0.51.2
```

.1.2 NAT

FW1
```
set nat source rule 10 outbound-interface eth3
set nat source rule 10 source address 10.0.0.0/8
set nat source rule 10 translation address 192.1.0.1-192.1.0.62
set nat source rule 20 outbound-interface eth4
set nat source rule 20 source address 10.0.0.0/8
set nat source rule 20 translation address 192.1.0.65-192.1.0.126
```

FW2
```
set nat source rule 10 outbound-interface eth3
set nat source rule 10 source address 10.0.0.0/8
set nat source rule 10 translation address 192.1.0.129-192.1.0.190
set nat source rule 20 outbound-interface eth4
set nat source rule 20 source address 10.0.0.0/8
set nat source rule 20 translation address 192.1.0.129-192.1.0.254
```

.1.3 Load Balancers Configuration

LB1

```
LB1 conntrack-sync
```

```
set high-availability vrrp group LB1Cluster vrid 10
set high-availability vrrp group LB1Cluster interface eth0
set high-availability vrrp group LB1Cluster virtual-address 192.168.100.1/24
set high-availability vrrp sync-group LB1Cluster member LB1Cluster
set high-availability vrrp group LB1Cluster rfc3768-compatibility

set service conntrack-sync accept-protocol 'tcp,udp,icmp'
set service conntrack-sync failover-mechanism vrrp sync-group LB1Cluster
set service conntrack-sync interface eth0
set service conntrack-sync mcast-group 225.0.0.50
set service conntrack-sync disable-external-cache
```

LB1A

```
set load-balancing wan interface-health eth1 nexthop 10.0.1.3
set load-balancing wan interface-health eth2 nexthop 10.0.2.4
set load-balancing wan rule 1 inbound-interface eth4
set load-balancing wan rule 1 interface eth1 weight 1
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan sticky-connections inbound
set load-balancing wan disable-source-nat
```

LB1B

```
set load-balancing wan interface-health eth1 nexthop 10.0.20.3
set load-balancing wan interface-health eth2 nexthop 10.0.3.4
set load-balancing wan rule 1 inbound-interface eth5
set load-balancing wan rule 1 interface eth1 weight 1
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan sticky-connections inbound
set load-balancing wan disable-source-nat
```

LB2

LB2 conntrack-sync

```
set high-availability vrrp group LB2Cluster vrid 20
set high-availability vrrp group LB2Cluster interface eth0
set high-availability vrrp group LB2Cluster virtual-address 192.168.200.1/24
set high-availability vrrp sync-group LB2Cluster member LB2Cluster
set high-availability vrrp group LB2Cluster rfc3768-compatibility

set service conntrack-sync accept-protocol 'tcp,udp,icmp'
set service conntrack-sync failover-mechanism vrrp sync-group LB2Cluster
set service conntrack-sync interface eth0
set service conntrack-sync mcast-group 225.0.0.100
set service conntrack-sync disable-external-cache
```

LB2A

```
set load-balancing wan interface-health eth1 nexthop 10.0.5.3
set load-balancing wan interface-health eth2 nexthop 10.0.11.4
set load-balancing wan rule 1 inbound-interface eth4
set load-balancing wan rule 1 interface eth1 weight 1
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan sticky-connections inbound
set load-balancing wan disable-source-nat
```

LB2B

```
set load-balancing wan interface-health eth1 nexthop 10.0.30.3
set load-balancing wan interface-health eth2 nexthop 10.0.4.4
set load-balancing wan rule 1 inbound-interface eth5
```

```
set load-balancing wan rule 1 interface eth1 weight 1
set load-balancing wan rule 1 interface eth2 weight 1
set load-balancing wan sticky-connections inbound
set load-balancing wan disable-source-nat
```

LB-DMZ

```
set load-balancing wan interface-health eth1 nexthop 10.0.50.1
set load-balancing wan interface-health eth2 nexthop 10.0.51.1
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 sticky-connections inbound
set load-balancing wan disable-source-nat
```

.1.4 Firewall Rules and Zones

Zones

```
set zone-policy zone INSIDE description "Inside (Internal Network)"
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 interface eth3
set zone-policy zone OUTSIDE interface eth4
set zone-policy zone DMZ description "DMZ (Server Farm)"
set zone-policy zone DMZ interface eth0
```

Firewalls and Rules

```
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 action accept
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 protocol udp
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 destination port 5000-6000
set zone-policy zone OUTSIDE from INSIDE firewall name FROM-INSIDE-TO-OUTSIDE

set firewall name FROM-OUTSIDE-TO-INSIDE rule 10 action accept
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 zone-policy zone INSIDE from OUTSIDE firewall name FROM-OUTSIDE-TO-INSIDE

set firewall name FROM-INSIDE-TO-DMZ rule 20 description "TCP Services"
set firewall name FROM-INSIDE-TO-DMZ rule 20 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 20 protocol tcp
set firewall name FROM-INSIDE-TO-DMZ rule 20 destination port 80,443,22
set firewall name FROM-INSIDE-TO-DMZ rule 20 destination address 192.1.1.100
set firewall name FROM-INSIDE-TO-DMZ rule 30 description "UDP Services"
set firewall name FROM-INSIDE-TO-DMZ rule 30 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 30 protocol udp
set firewall name FROM-INSIDE-TO-DMZ rule 30 destination port 53
set firewall name FROM-INSIDE-TO-DMZ rule 30 destination address 192.1.1.101
set zone-policy zone DMZ from INSIDE firewall name FROM-INSIDE-TO-DMZ

set firewall name FROM-DMZ-TO-INSIDE rule 10 action accept
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 zone-policy zone INSIDE from DMZ firewall name FROM-DMZ-TO-INSIDE

set firewall name FROM-OUTSIDE-TO-DMZ rule 20 description "TCP Services"
```

```
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 action accept
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 protocol tcp
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 destination port 80,443
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 destination address 192.1.1.100
set firewall name FROM-OUTSIDE-TO-DMZ rule 30 description "UDP Services"
set firewall name FROM-OUTSIDE-TO-DMZ rule 30 action accept
set firewall name FROM-OUTSIDE-TO-DMZ rule 30 protocol udp
set firewall name FROM-OUTSIDE-TO-DMZ rule 30 destination port 53
set firewall name FROM-OUTSIDE-TO-DMZ rule 30 destination address 192.1.1.101
set zone-policy zone DMZ from OUTSIDE firewall name FROM-OUTSIDE-TO-DMZ

set firewall name FROM-DMZ-TO-OUTSIDE rule 10 action accept
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 zone-policy zone OUTSIDE from DMZ firewall name FROM-DMZ-TO-OUTSIDE
```

.1.5 Python Script to apply blocking rules to Firewalls

```
#!/usr/bin/env python3

from netmiko import ConnectHandler

fw1 = {
  "device_type": "vyos",
  "host": "10.0.50.1",
  "username": "vyos",
  "password": "vyos",
  "port": 22,
}

fw2 = {
    "device_type": "vyos",
    "host": "10.0.51.1",
    "username": "vyos",
    "password": "vyos",
    "port": 22,
}

net_connect_fw1 = ConnectHandler(**fw1)
net_connect_fw2 = ConnectHandler(**fw2)

# Identify the attackers' IP addresses
attackers_ips = ['200.2.2.200']

# Create a list of firewall blocking rules for the attackers' IP addresses
block_rules = []
i=1
for ip in attackers_ips:
    block_rule = f'set firewall name FROM-OUTSIDE-TO-INSIDE rule '
+ str(i) +' source address '+str(ip)+'/32'
    block_rules.append(block_rule)
    block_rule = f'set firewall name FROM-OUTSIDE-TO-INSIDE rule '
+ str(i) +' action drop'
    block_rules.append(block_rule)
    block_rule = f'set firewall name FROM-OUTSIDE-TO-DMZ rule '
+ str(i) +' source address '+str(ip)+'/32'
    block_rules.append(block_rule)
```

```
    block_rule = f'set firewall name FROM-OUTSIDE-TO-DMZ rule '
+ str(i) +' action drop'
    block_rules.append(block_rule)
    i=i+1
    if i%10==0: i=i+1

# Set configuration and create blocking rules
config_commands = [
                    *block_rules,
                  ]

print("\n----FW1----\n")
output_fw1 = net_connect_fw1.send_config_set(config_commands, exit_config_mode=False)
print(output_fw1)
output_fw1 = net_connect_fw1.commit()
print(output_fw1)
output_fw1 = net_connect_fw1.send_command("show firewall")
print(output_fw1)

print("\n----FW2----\n")
output_fw2 = net_connect_fw2.send_config_set(config_commands, exit_config_mode=False)
print(output_fw2)
output_fw2 = net_connect_fw2.commit()
print(output_fw2)
output_fw2 = net_connect_fw2.send_command("show firewall")
print(output_fw2)
```