

# SRC - Projecto 1

Universidade de Aveiro  
DETI

Tiago Santos, (89356), Rodrigo Rodrigues (102573)



# Índice

<b>1</b>	<b>Topologia e configurações</b>	<b>2</b>
<b>2</b>	<b>Load Balancing</b>	<b>3</b>
2.1	Configuração dos Load Balancers . . . . .	4
<b>3</b>	<b>Firewalls</b>	<b>5</b>
3.1	Serviços e regras . . . . .	7
3.2	DDoS attack Script . . . . .	12

# 1 Topologia e configurações

Foi estruturada uma topologia de rede que demonstra um cenário que faz uso de firewall de alta disponibilidade. Dividimos a rede em três zonas: INSIDE, OUTSIDE, DMZ, criando assim limites lógicos entre as diferentes partes da rede e aplicando políticas específicas a cada zona para controlo de tráfego. Entre estas zonas encontram-se load balancers e Firewalls para garantir uma distribuição equitativa de carga e reforçar a segurança da rede.

No INSIDE (Internal Network) contém a subrede 10.2.2.0/24 que contém o VPC e faz a ligação aos loadbalancers através da subrede 10.1.1.0/24.

Para o OUTSIDE (Internet) foi considerada a subrede 200.2.2.0/24 onde colocámos um VPC com o objectivo de simular um dispositivo externo que queremos aceder a partir da empresa e uma VM para tornar servir como servidor DNS.

Por fim, o DMZ contém a subrede 192.1.1.0 e está diretamente ligada, por um switch, a ambas as firewalls. Foram colocados dois dispositivos um VPC (192.1.1.100) e uma VM (192.1.1.200), VPC para facilmente testar a distribuiçao de pacotes icmp e udp, e VM para os pacotes tcp.( e correr o script.)

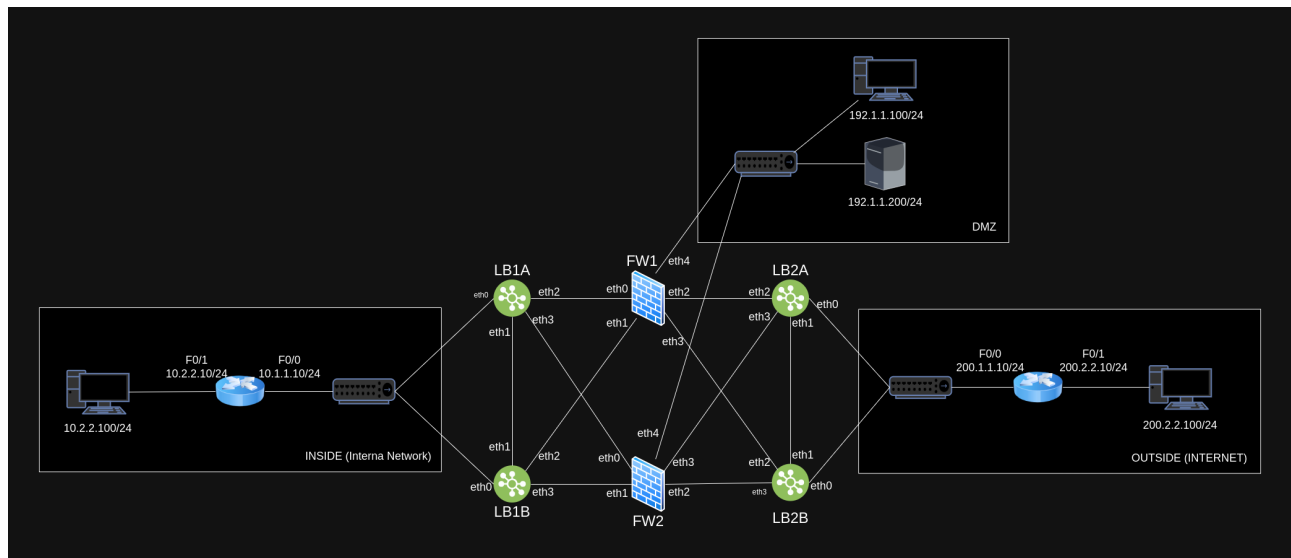


Figura 1: Vista global do sistema

IPs atribuídos a entradas						
IPs	LB1A	LB1B	LB2A	LB2B	FW1	FW2
eth0	10.1.1.1	10.1.1.2	200.1.1.1	200.1.1.2	10.1.3.2	10.1.4.2
eth1	10.1.2.1	10.1.2.2	10.5.0.1	10.5.0.2	10.2.0.2	10.2.1.2
eth2	10.1.3.1	10.2.1.1	10.3.0.2	10.3.1.2	10.3.0.1	10.4.0.1
eth3	10.4.1	10.2.2.1	10.4.1.2	10.4.0.2	10.3.1.1	10.4.1.1
eth4	x	x	x	x	192.1.1.1.	192.1.1.2.

Tabela 1: Obtained compression for the sample files

## 2 Load Balancing

É o processo de distribuir o trabalho entre vários recursos que podem computar essa carga. O mesmo conceito se aplica às firewalls, onde o tráfego é distribuído por várias firewalls para garantir um processamento eficiente. Firewalls com load balancing têm diversas vantagens. A melhoria de desempenho é uma consequência da distribuição de trabalho, evitando que uma firewall se torne um bottleneck. Com o load balancing, também se obtém maior disponibilidade. Se uma firewall falhar, o load balance consegue redirecionar o tráfego para uma firewall de backup, garantindo a continuidade do processamento do tráfego. A redundância e a segurança resultantes da aplicação de políticas distintas a diferentes segmentos de redes são o foco da nossa análise. Nas nossas firewalls, são aplicadas regras ao fluxo geral de comunicações, não apenas aos pacotes individuais. Ao analisar a implementação do load balancing na nossa rede, percebemos que os load balancers de carga escolhem a interface para enviar um pacote com um destino específico, direcionando todos os pacotes subsequentes com a mesma origem e destino através dessa mesma interface. Isso elimina a necessidade de sincronização entre as firewalls, pois cada fonte envia sempre para o mesmo destino, passando pela mesma firewall devido ao balanceamento de carga implementado.

Nesta seção, é relevante mencionar que existem algoritmos de load balancing de carga que determinam qual servidor/firewall receberá uma solicitação específica. Um exemplo é o IP Hash, que usa os IPs de origem e destino dos pacotes para selecionar o destino do pedido por meio de funções de hash. Este algoritmo não requer sincronização de estados, nem no firewall nem no load balancer, pois cada balanceador consegue calcular independentemente o destino de cada solicitação. Por outro lado, algoritmos como o Round-Robin ou Least Connections exigem que os load balancers mantenham um estado compartilhado, o que pode resultar em problemas de sincronização.

Na nossa topologia, os load balancers são stateful, pois mantêm o estado das rotas escolhidas e o compartilham e sincronizam com outros load balancers. Considerando o algoritmo de load balancing utilizado (round-robin), é relevante discutir situações em que manter essa sincronização pode ser prejudicial, como em ataques DDOS. Durante um ataque desse tipo, a sincronização torna-se problemática devido ao alto consumo de recursos. Sincronizar estados

de dispositivos e conexões em redes com vários nós requer muita memória, o que pode ser explorado num ataque DDOS, criando um bottleneck e reduzindo o desempenho do sistema. Além disso, se o mecanismo de sincronização não conseguir lidar com grandes volumes de tráfego, ele pode se tornar um alvo para os atacantes ampliarem o ataque. Da mesma forma, se ocorrer um erro de sincronização devido ao tráfego intenso, o sistema pode ficar mais vulnerável ao ataque, caso não consiga recuperar rapidamente o suficiente para continuar a funcionar.

## 2.1 Configuração dos Load Balancers

Foram criadas rotas estáticas nos Load Balancers para as pontas. O LB1A e LB1B têm rotas que direcionam o tráfego para o **10.2.2.0/24** pelo **10.1.1.10** (Router interno), enquanto que nos LB2A e LB2B o tráfego para o **200.2.2.0/24** é redirecionado para o **200.1.1.10** (Router externo).

Para LB1A e LB1B:

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

Para LB2A e LB2B:

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

Para além disso os Load Balancers têm o conntrack-sync configurado na porta **eth1** com o Load Balancing nas portas **eth2** e **eth3** ambas com peso 1, garantindo uma distribuição de tráfego 50/50.

É ainda importante mencionar que as interfaces de cada Load Balancer ligados à firewall são as mesmas. O facto de o conntrack-sync guardar os pacotes que o Load Balancer recebe e em que portas, assim se um pacote que passou pelo LB2A pode ser redirecionado pelo LB2B não vai ter problemas em chegar ao destino.

```
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
617032258 10.2.2.100:28089 200.2.2.100:7    udp [17]    23
3360640403 10.2.2.100:10639 200.2.2.100:7    udp [17]    142
1852991575 10.1.2.2:46398   225.0.0.50:3780  udp [17]    29
vyos@LB1A:~$
vyos@LB1A:~$ show wan-load-balance status
Chain WANLOADBALANCE_PRE (1 references)
pkts bytes target prot opt in out source destination state NEW statistic mode random probability 0.5000000000
23 1932 ISP_eth2 all -- eth0 * 0.0.0.0/0 0.0.0.0/0 state NEW
22 1848 ISP_eth3 all -- eth0 * 0.0.0.0/0 0.0.0.0/0 state NEW
0 0 CONNMARK all -- eth0 * 0.0.0.0/0 0.0.0.0/0 CONNMARK restore
vyos@LB1A:~$
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
2492130795 10.2.2.100:59343 200.2.2.100:7    udp [17]    279
907885040 10.1.2.1         224.0.0.18       vrrp [112]   599
2019577093 10.2.2.100:7297 200.2.2.100:7    udp [17]    157
73031358 10.2.2.100:10639 200.2.2.100:7    udp [17]    42
vyos@LB1B:~$
vyos@LB1B:~$ show wan-load-balance status
Chain WANLOADBALANCE_PRE (1 references)
pkts bytes target prot opt in out source destination state NEW statistic mode random probability 0.5000000000
255 21400 ISP_eth2 all -- eth0 * 0.0.0.0/0 0.0.0.0/0 state NEW
265 22260 ISP_eth3 all -- eth0 * 0.0.0.0/0 0.0.0.0/0 state NEW
656 55104 CONNMARK all -- eth0 * 0.0.0.0/0 0.0.0.0/0 CONNMARK restore
vyos@LB1B:~$
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
2148247419 10.5.0.1:52370   225.0.0.50:3780  udp [17]    29
1494344711 192.1.0.23:43357 200.2.2.100:7    udp [17]    113
3457457692 10.5.0.1         224.0.0.18       vrrp [112]   599
156249789 192.1.0.23:59343 200.2.2.100:7    udp [17]    9
3852819567 192.1.0.13:46205 200.2.2.100:7    udp [17]    179
vyos@LB2A:~$
vyos@LB2A:~$ show wan-load-balance status
Chain WANLOADBALANCE_PRE (1 references)
pkts bytes target prot opt in out source destination state NEW statistic mode random probability 0.5000000000
0 0 ISP_eth2 all -- eth0 * 0.0.0.0/0 0.0.0.0/0 state NEW
0 0 ISP_eth3 all -- eth0 * 0.0.0.0/0 0.0.0.0/0 state NEW
1366 115K CONNMARK all -- eth0 * 0.0.0.0/0 0.0.0.0/0 CONNMARK restore
vyos@LB2A:~$
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
2148247419 10.5.0.1:52370   225.0.0.50:3780  udp [17]    29
3457457692 10.5.0.1         224.0.0.18       vrrp [112]   599
3852819567 192.1.0.13:46205 200.2.2.100:7    udp [17]    66
3657071340 192.1.0.13:3164  200.2.2.100:7    udp [17]    179
vyos@LB2B:~$
vyos@LB2B:~$ show wan-load-balance status
Chain WANLOADBALANCE_PRE (1 references)
pkts bytes target prot opt in out source destination state NEW statistic mode random probability 0.5000000000
0 0 ISP_eth2 all -- eth0 * 0.0.0.0/0 0.0.0.0/0 state NEW
0 0 ISP_eth3 all -- eth0 * 0.0.0.0/0 0.0.0.0/0 state NEW
1454 122K CONNMARK all -- eth0 * 0.0.0.0/0 0.0.0.0/0 CONNMARK restore
vyos@LB2B:~$
```

Figura 2: Resultados dos comandos "conntrack table ipv4" e "wan-load-balance status" em cada Load Balancer

### 3 Firewalls

Para a firewall configuramos rotas estáticas, todo o tráfego vai para o next-hop ligado ao eth2 e eth3. Tráfego para o 10.2.2.0/24 vai para o next-hop eth0 e eth1.

FW1:

```
set protocols static route 0.0.0.0/0 next-hop 10.3.0.2
set protocols static route 0.0.0.0/0 next-hop 10.3.1.2
set protocols static route 10.2.2.0/24 next-hop 10.1.3.1
set protocols static route 10.2.2.0/24 next-hop 10.2.0.1
```

FW2:

```
set protocols static route 0.0.0.0/0 next-hop 10.4.1.2 (ip eth3
LB2A)
set protocols static route 0.0.0.0/0 next-hop 10.4.0.2 (ip eth3
LB2B)
set protocols static route 10.2.2.0/24 next-hop 10.1.4.1 (ip eth3
LB1A)
set protocols static route 10.2.2.0/24 next-hop 10.2.1.1 (ip eth3
LB1B)
```

Para as traduções NAT definimos 4 pools, 2 para cada Firewall, para garantirmos que não há ips-portas repetidas:

```
vyos@FW1:~$ show nat source translations
Pre-NAT      Post-NAT      Prot  Timeout
10.1.3.1      10.1.3.1      icmp  30
10.3.1.2      10.3.1.2      icmp  29
10.2.2.100    192.1.0.13    udp   179
10.2.0.1      10.2.0.1      icmp  30
10.3.0.2      10.3.0.2      icmp  25
vyos@FW1:~$
```

```
vyos@FW2:~$ show nat source translations
Pre-NAT      Post-NAT      Prot  Timeout
10.2.1.1      10.2.1.1      icmp  27
10.2.2.100    192.1.0.23    udp   179
10.1.4.1      10.1.4.1      icmp  27
10.4.1.2      10.4.1.2      icmp  27
10.4.0.2      10.4.0.2      icmp  26
vyos@FW2:~$
```

Figura 3: Traduções feitas por NAT em cada Firewall

```
vyos@FW1:~$ show nat source translations
Pre-NAT      Post-NAT      Prot  Timeout
10.1.3.1      10.1.3.1      icmp  30
10.3.1.2      10.3.1.2      icmp  29
10.2.2.100    192.1.0.13    udp   179
10.2.0.1      10.2.0.1      icmp  30
10.3.0.2      10.3.0.2      icmp  25
vyos@FW1:~$
```

```
vyos@FW2:~$ show nat source translations
Pre-NAT      Post-NAT      Prot  Timeout
10.2.1.1      10.2.1.1      icmp  27
10.2.2.100    192.1.0.23    udp   179
10.1.4.1      10.1.4.1      icmp  27
10.4.1.2      10.4.1.2      icmp  27
10.4.0.2      10.4.0.2      icmp  26
vyos@FW2:~$
```

Figura 4: Antes e depois das traduções

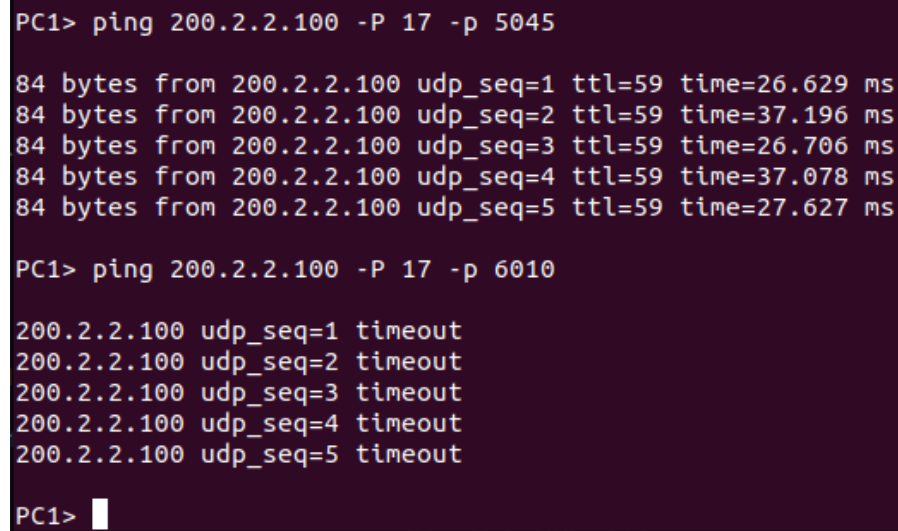
### 3.1 Serviços e regras

Começamos por definir as zonas:

```
#Para a FW1
set zone-policy zone INSIDE description "Inside (Internal Network)"
set zone-policy zone INSIDE interface eth0
set zone-policy zone INSIDE interface eth1
set zone-policy zone DMZ description "DMZ (Server Farm)"
set zone-policy zone DMZ interface eth4
set zone-policy zone OUTSIDE description "Outside (Internet)"
set zone-policy zone OUTSIDE interface eth2
set zone-policy zone OUTSIDE interface eth3
```

Foram implementadas diversas regras nas firewalls. Começamos por meter uma regra que indica que a zona INSIDE pode comunicar com a OUTSIDE usando UDP apenas nas portas 5000 a 6000. Isto poderia ser útil por exemplo para a empresa permitir comunicação específica ao usar um software de comunicação em tempo real com clientes fora da rede da empresa.

```
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 action accept
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 description "
streaming ports access 5000-6000"
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 protocol udp
set firewall name FROM-INSIDE-TO-OUTSIDE rule 10 destination port
5000-6000
```



```
PC1> ping 200.2.2.100 -P 17 -p 5045

84 bytes from 200.2.2.100 udp_seq=1 ttl=59 time=26.629 ms
84 bytes from 200.2.2.100 udp_seq=2 ttl=59 time=37.196 ms
84 bytes from 200.2.2.100 udp_seq=3 ttl=59 time=26.706 ms
84 bytes from 200.2.2.100 udp_seq=4 ttl=59 time=37.078 ms
84 bytes from 200.2.2.100 udp_seq=5 ttl=59 time=27.627 ms

PC1> ping 200.2.2.100 -P 17 -p 6010

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> █
```

Figura 5: Resultados do envio de pacotes UDP nas portas 5045 e 6010

Também de INSIDE para OUTSIDE colocamos uma regra que permite ligações tcp para acesso a serviços HTTP e HTTPS.



```

set firewall name FROM-INSIDE-TO-OUTSIDE rule 11 description "
Inside network access for HTTP and HTTPS through TCP"
set firewall name FROM-INSIDE-TO-OUTSIDE rule 11 action accept
set firewall name FROM-INSIDE-TO-OUTSIDE rule 11 protocol tcp
set firewall name FROM-INSIDE-TO-OUTSIDE rule 11 destination port
80,443

```

De seguida colocámo uma regras que impeçam o DMZ e o OUTSIDE de começar qualquer comunicação com o INSIDE:

```

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

```

A mesma regra foi também implementada entre o OUTSIDE e o INSIDE para que o primeiro não possa iniciar comunicação com o segundo:

```

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

```

Depois implementámos 2 regras entre INSIDE e DMZ. Uma que aceita pacotes ICMP e outra pacotes tcp:

```

set firewall name FROM-INSIDE-TO-DMZ rule 10 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 10 description "Accept
ICMP packets 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 12 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 12 description "Accept
HTTP and HTTPS from INSIDE to DMZ"
set firewall name FROM-INSIDE-TO-DMZ rule 12 destination address
192.1.1.0/24
set firewall name FROM-INSIDE-TO-DMZ rule 12 destination port
80,443
set firewall name FROM-INSIDE-TO-DMZ rule 12 protocol tcp

```

```

PC1> ping 192.1.1.100

84 bytes from 192.1.1.100 icmp_seq=1 ttl=61 time=14.322 ms
192.1.1.100 icmp_seq=2 timeout
84 bytes from 192.1.1.100 icmp_seq=3 ttl=61 time=12.373 ms
84 bytes from 192.1.1.100 icmp_seq=4 ttl=61 time=16.564 ms
84 bytes from 192.1.1.100 icmp_seq=5 ttl=61 time=16.922 ms

PC1> █

```

Figura 6: Resultados do envio de pacotes ICMP para DMZ

```

PC1> ping 192.1.1.200 -P 6 -p 443

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

PC1> █

```

Figura 7: Resultados do envio de pacotes TCP para DMZ

Para permitir acesso ao servidor DNS, colocámos uma regra que permite a comunicação de pacotes TCP pela porta 53.

```

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

```

```

PC2> ping 192.1.1.100 -p 53 -P 17

84 bytes from 192.1.1.100 udp_seq=1 ttl=61 time=11.278 ms
84 bytes from 192.1.1.100 udp_seq=2 ttl=61 time=16.023 ms
84 bytes from 192.1.1.100 udp_seq=3 ttl=61 time=15.138 ms
84 bytes from 192.1.1.100 udp_seq=4 ttl=61 time=15.700 ms
84 bytes from 192.1.1.100 udp_seq=5 ttl=61 time=16.367 ms

PC2> █

```

Figura 8

Por ultimo, para permitir acesso a pacotes HTTP, HTTPS e ssh criamos uma regra para permitir acesso a pacotes TCP na DMZ.

```

set firewall name FROM-INSIDE-TO-DMZ rule 15 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 15 description "Accept
HTTP, HTTPS and SSH from INSIDE to DMZ"
set firewall name FROM-INSIDE-TO-DMZ rule 15 destination address
192.1.1.0/24
set firewall name FROM-INSIDE-TO-DMZ rule 15 destination port
22,80,443
set firewall name FROM-INSIDE-TO-DMZ rule 15 protocol tcp
set firewall name FROM-INSIDE-TO-DMZ rule 15 source address
10.2.2.0/2

```

Resumindo temos aqui a lista das regras implementadas em cada firewall:

```

name FROM-DMZ-TO-INSIDE {
    default-action drop
    rule 10 {
        action accept
        description "Accept established-related connections"
        state {
            established enable
            related enable
        }
    }
}
name FROM-DMZ-TO-OUTSIDE {
    default-action drop
    rule 10 {
        action accept
        description "Accept Established-related connections"
        state {
            established enable
            related enable
        }
    }
}
name FROM-INSIDE-TO-DMZ {
    default-action drop
    rule 10 {
        action accept

```

```

        description "Accept ICMP packets to DMZ"
        destination {
            address 192.1.1.0/24
        }
        icmp {
            type 8
        }
        protocol icmp
    }
    rule 15 {
        action accept
        description "Accept HTTP, HTTPS and SSH from INSIDE to DMZ"
        destination {
            address 192.1.1.0/24
            port 22,80,443
        }
        protocol tcp
        source {
            address 10.2.2.0/2
        }
    }
}
name FROM-INSIDE-TO-OUTSIDE {
    default-action drop
    rule 10 {
        action accept
        description "Accept UDP in ports 5000-6000"
        destination {
            port 5000-6000
        }
        protocol udp
    }
    rule 11 {
        action accept
        description "inside network access for HTTP and HTTPS
through TCP"
        destination {
            port 80,443
        }
        protocol tcp
    }
}
name FROM-OUTSIDE-TO-DMZ {
    default-action drop
    rule 12 {
        action accept
        description "outside access to external dns"
        destination {
            address 192.1.1.100
            port 53
        }
        protocol udp
        source {
            address !10.2.2.0/24
        }
    }
}

```

```

}
name FROM-OUTSIDE-TO-INSIDE {
    default-action drop
    rule 10 {
        action accept
        description "Accept Established-related connections"
        state {
            established enable
            related enable
        }
    }
}
}

```

### 3.2 DDoS attack Script

Para simular uma defesa eficaz contra ataques de DoS e DDoS, desenvolvemos um script Python que pode ser executado em um servidor localizado na DMZ. Esse script é projetado para atualizar dinamicamente uma lista de IPs que serão bloqueados pelas firewalls, fornecendo uma camada adicional de proteção contra ameaças externas.

Primeiro foi criado uma **address-group** vazia nas firewalls:

```
set firewall group address-group BLOCK_ATK
```

De seguida implementámos uma regra que consiga bloquear os IPs que estejam nesse grupo :

```

set firewall name FROM-OUTSIDE-TO-DMZ rule 20 description "Block
attackers"
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 action drop
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 source BLOCK_ATK
set firewall name FROM-OUTSIDE-TO-DMZ rule 20 protocol all

set firewall name FROM-OUTSIDE-TO-INSIDE rule 20 description "Block
attackers"
set firewall name FROM-OUTSIDE-TO-INSIDE rule 20 action drop
set firewall name FROM-OUTSIDE-TO-INSIDE rule 20 source BLOCK_ATK
set firewall name FROM-OUTSIDE-TO-INSIDE rule 20 protocol all

```

Depois de criar a lista BLOCK\_ATK e atribuí-la a ambas as regras de firewall, para garantir sua atualização eficiente, desenvolvemos um script Python que utiliza a biblioteca "vymgmt". Essa biblioteca é especialmente útil para interagir com dispositivos VyOS de maneira programática, permitindo automatizar tarefas de configuração.

O script Python desenvolvido facilita o processo de atualização das regras de firewall. Ao realizar o login nas firewalls, ele entra no modo de configuração e atualiza a lista BLOCK\_ATK, garantindo que ambas as regras façam referência à versão mais recente dessa lista.

Essa abordagem automatizada simplifica o gerenciamento de firewall, tornando-o mais ágil e menos propenso a erros humanos. Além disso, ao utilizar o script Python, podemos integrar facilmente a atualização da lista BLOCK\_ATK em fluxos de trabalho mais amplos e sistemas de monitoramento de segurança.

Ficheiro python a ser executado:

```
from vyangmt import Router
import re

def main():
    FW1_router_ip = '192.1.1.1'
    FW2_router_ip = '192.1.1.2'
    username = 'vyos'
    password = 'vyos'
    attackers = ['192.1.1.54', '192.1.1.56', '192.1.1.108']

    router = Router(FW1_router_ip, username, password)
    try:
        router.login()
        router.configure()
        for attacker in attackers:
            router.set(f'firewall group address-group BLOCK_ATK
address {attacker};')
        router.commit()
        router.save()

    except Exception as e:
        print(f"An error occurred: {e}")
    finally:
        router.close()

    router = Router(FW2_router_ip, username, password)
    try:
        router.login()
        router.configure()
        for attacker in attackers:
            router.set(f'firewall group address-group BLOCK_ATK
address {attacker};')
        router.commit()
        router.save()

    except Exception as e:
        print(f"An error occurred: {e}")
    finally:
        router.close()

if __name__ == "__main__":
    main()
```