Projeto 1 – SRC

Departamento de Eletrónica, Telecomunicações e Informática

Universidade de Aveiro

Diogo Torrinhas (98440), João Torrinhas (98435)

16 de abril 2023

Índice

Exercício 9

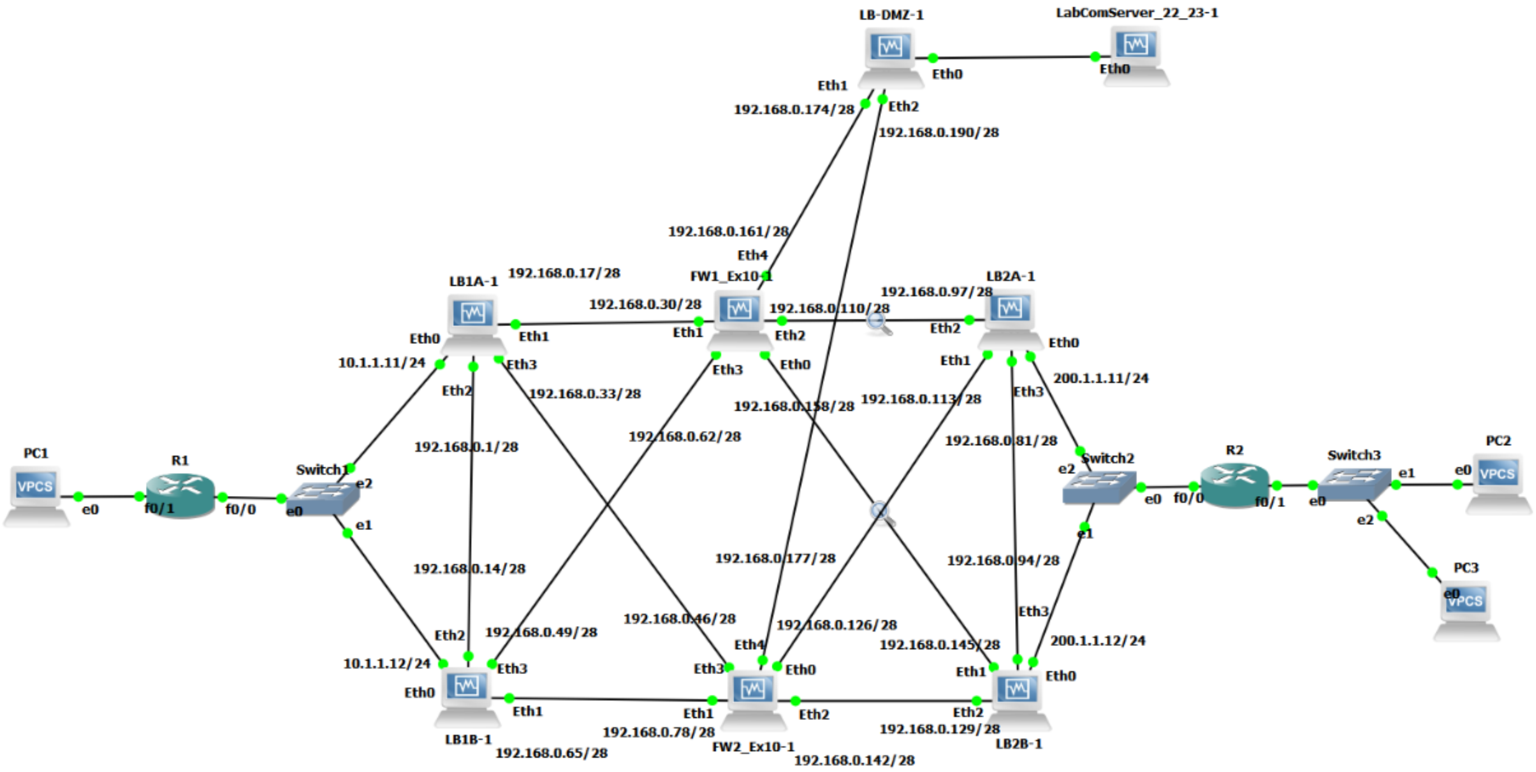


Figura 1-Rede

Neste exercício dividimos a rede 192.168.0.0/24 em várias subredes como mostra a figura 1.

Nas figuras 2 e 3 estão presentes as configurações para um Load Balancer e para uma firewall, uma vez que para os restantes Load Balancers e a outra firewall as configurações são semelhantes, pois apenas mudam endereços ip e algumas rotas estáticas. De salientar que na configuração dos Load Balancers, apesar de nós termos configurado rotas estáticas para as firewalls, não era necessário por causa do comando *set load-balancing wan.*

```
EXERCICIO 9:


---------ROUTER INSIDE-----------
conf term
ip route 0.0.0.0 0.0.0.0 10.1.1.12
ip route 0.0.0.0 0.0.0.0 10.1.1.11
end
write

---------ROUTER OUTSIDE-----------
conf term
ip route 192.1.0.0 255.255.254.0 200.1.1.11
ip route 192.1.0.0 255.255.254.0 200.1.1.12
ip route 10.0.0.0 255.0.0.0 200.1.1.11
end
write


---------LB1B-----------
!Forçar o ping do inside a ir por baixo
configure
set protocols static route 10.2.2.0/24 next-hop 10.1.1.10
set protocols static route 0.0.0.0/0 next-hop 192.168.0.62 !N é preciso
set protocols static route 0.0.0.0/0 next-hop 192.168.0.78 !N é preciso
commit
save

set load-balancing wan interface-health eth1 nexthop 192.168.0.78
set load-balancing wan interface-health eth3 nexthop 192.168.0.62
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 eth3 weight 1
set load-balancing wan sticky-connections inbound
set load-balancing wan disable-source-nat
commit
save

set high-availability vrrp group FWCluster vrid 10
set high-availability vrrp group FWCluster interface eth2
set high-availability vrrp group FWCluster virtual-address 192.168.100.1/24
set high-availability vrrp sync-group FWCluster member FWCluster
set high-availability vrrp group FWCluster rfc3768-compatibility
commit
save

set service conntrack-sync accept-protocol 'tcp,udp,icmp'
set service conntrack-sync failover-mechanism vrrp sync-group FWCluster
set service conntrack-sync interface eth2
set service conntrack-sync mcast-group 225.0.0.50
set service conntrack-sync disable-external-cache
commit
save
```

Figura 2-Configurações routers e LB1B

```
---------FW1---------
configure
set protocols static route 0.0.0.0/0 next-hop 192.168.0.97
set protocols static route 10.2.2.0/24 next-hop 192.168.0.17
set protocols static route 10.2.2.0/24 next-hop 192.168.0.49 !
commit
save

set nat source rule 20 outbound-interface eth0
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 nat source rule 10 outbound-interface eth2
set nat source rule 10 source address 10.0.0.0/8 !192.168.0.0/8
set nat source rule 10 translation address 192.1.0.1-192.1.0.10
commit
save

set zone-policy zone INSIDE description "Inside (Internal Network)"
set zone-policy zone INSIDE interface eth1
set zone-policy zone INSIDE interface eth3
set zone-policy zone OUTSIDE description "Outside (Internet)"
set zone-policy zone OUTSIDE interface eth0
set zone-policy zone OUTSIDE interface eth2

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 firewall name FROM-OUTSIDE-TO-INSIDE rule 10 action accept
set firewall name FROM-OUTSIDE-TO-INSIDE rule 10 state established enable

set zone-policy zone INSIDE from OUTSIDE firewall name FROM-OUTSIDE-TO-INSIDE
set zone-policy zone OUTSIDE from INSIDE firewall name FROM-INSIDE-TO-OUTSIDE
commit
save
```

Figura 3-Configurações FW1

Exercício 9.1

Através dos comandos set *load-balancing wan sticky-connections inbound* e *set load-balancing wan disable-source-nat* permite que não haja a necessidade de as Firewalls terem sincronização. Com o comando *set load-balancing wan sticky-connections* as interfaces vão memorizar de onde o tráfego foi recebido permitindo que o tráfego entre dispositivos seja sempre enviado pela mesma ligação e, com o comando *set load-balancing wan disable-source-nat*, o endereço ip de origem dos pacotes de saída será preservado quando forem enviados pela interface WAN de maneira a garantir que os pacotes sejam roteados corretamente de volta ao remetente. Por exemplo, se o request do ping do pc1 for por LB1A-FW1-LB2A até ao pc2, o mesmo, por causa dos comandos anteriores, vai saber por onde tem de enviar o reply para o pc1.

Exercício 9.2

O algoritmo IP Hash pode permitir a inexistência de sincronização em Load Balancers porque, neste algoritmo uma hash function é usada para mapear cada request para um servidor específico. A hash function assegura que o mesmo request é enviado para o mesmo servidor, assim, cada load balancer, através do endereço ip do cliente, pode calcular, independentemente, o servidor/firewall para qual o request dever ser enviado, eliminando a necessidade de sincronização.

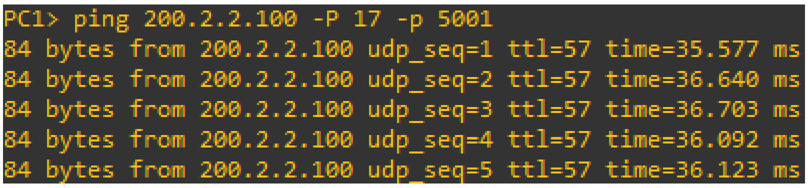
Exercício 9.3

Um ataque DDOS é quando o atacante tenta sobrecarregar um servidor/sistema com um elevado número de pedidos/pacotes tornando esse mesmo sistema inválido ou extremamente lento devido à sobrecarga de recursos. Os dispositivos que tenham a sincronização ativa estão constantemente a trocar informação para se atualizarem. Se houver um ataque DDOS nesses dispositivos de sincronização, a rede pode tornar-se muito mais lenta, pelo número elevado de pacotes a serem encaminhados nas interfaces de sincronização, por exemplo, se houver um ataque DDOS no LB1A, como o LB1A e o LB1B estão sincronizados, esse ataque ia se propagar para o LB1B, ou seja, ia afetar ainda mais a rede levando a um efeito em cascada. Posto isto, a sincronização pode ser prejudicial durante um ataque de DDOS.

Resultados

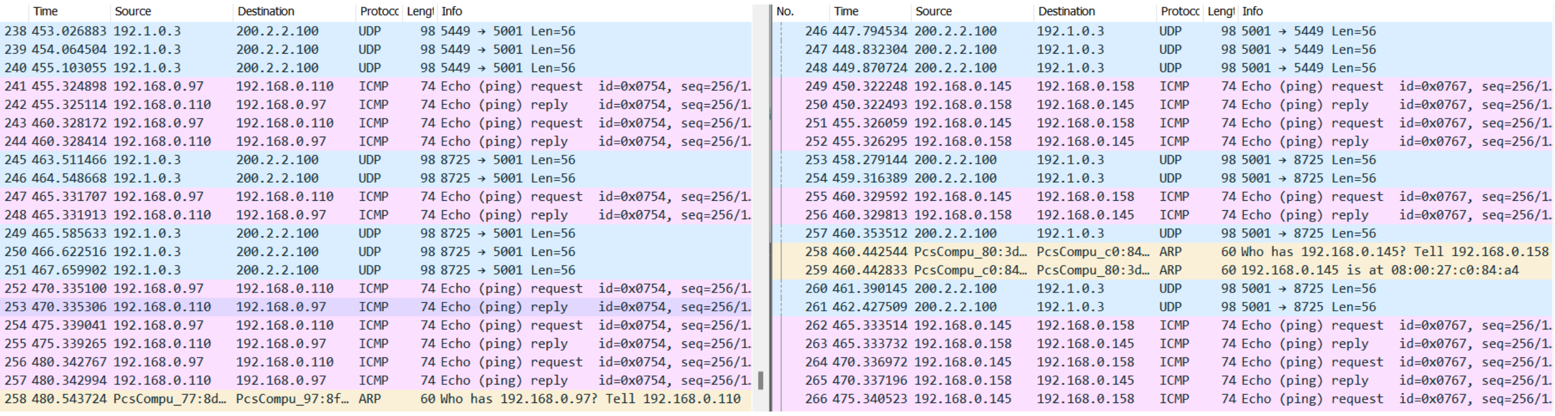
Testámos os pings do inside para o outside e verificámos que o ping foi bem-sucedido como se pode ver na figura 4.

Como se pode ver no print em baixo, com as capturas do wireshark (na figura 1 com a imagem da rede dá para ver onde estão a ser feitas estas capturas), o ping passou com sucesso nas firewalls e ainda conseguimos ver a translação dos ip´s na tabela NAT (o endereço ip de origem 10.2.2.100 transformou-se em 192.1.0.3).



```
PC1> ping 200.2.2.100 -P 17 -p 5001
84 bytes from 200.2.2.100 udp_seq=1 ttl=57 time=35.577 ms
84 bytes from 200.2.2.100 udp_seq=2 ttl=57 time=36.640 ms
84 bytes from 200.2.2.100 udp_seq=3 ttl=57 time=36.703 ms
84 bytes from 200.2.2.100 udp_seq=4 ttl=57 time=36.092 ms
84 bytes from 200.2.2.100 udp_seq=5 ttl=57 time=36.123 ms
```

Figura 4-Ping inside para Outside



	Time	Source	Destination	Protocc	Lengt	Info
238	453.026883	192.1.0.3	200.2.2.100	UDP	98	5449 → 5001 Len=56
239	454.064504	192.1.0.3	200.2.2.100	UDP	98	5449 → 5001 Len=56
240	455.103055	192.1.0.3	200.2.2.100	UDP	98	5449 → 5001 Len=56
241	455.324898	192.168.0.97	192.168.0.110	ICMP	74	Echo (ping) request id=0x0754, seq=256/1.
242	455.325114	192.168.0.110	192.168.0.97	ICMP	74	Echo (ping) reply id=0x0754, seq=256/1.
243	460.328172	192.168.0.97	192.168.0.110	ICMP	74	Echo (ping) request id=0x0754, seq=256/1.
244	460.328414	192.168.0.110	192.168.0.97	ICMP	74	Echo (ping) reply id=0x0754, seq=256/1.
245	463.511466	192.1.0.3	200.2.2.100	UDP	98	8725 → 5001 Len=56
246	464.548668	192.1.0.3	200.2.2.100	UDP	98	8725 → 5001 Len=56
247	465.331707	192.168.0.97	192.168.0.110	ICMP	74	Echo (ping) request id=0x0754, seq=256/1.
248	465.331913	192.168.0.110	192.168.0.97	ICMP	74	Echo (ping) reply id=0x0754, seq=256/1.
249	465.585633	192.1.0.3	200.2.2.100	UDP	98	8725 → 5001 Len=56
250	466.622516	192.1.0.3	200.2.2.100	UDP	98	8725 → 5001 Len=56
251	467.659902	192.1.0.3	200.2.2.100	UDP	98	8725 → 5001 Len=56
252	470.335100	192.168.0.97	192.168.0.110	ICMP	74	Echo (ping) request id=0x0754, seq=256/1.
253	470.335306	192.168.0.110	192.168.0.97	ICMP	74	Echo (ping) reply id=0x0754, seq=256/1.
254	475.339041	192.168.0.97	192.168.0.110	ICMP	74	Echo (ping) request id=0x0754, seq=256/1.
255	475.339265	192.168.0.110	192.168.0.97	ICMP	74	Echo (ping) reply id=0x0754, seq=256/1.
256	480.342767	192.168.0.97	192.168.0.110	ICMP	74	Echo (ping) request id=0x0754, seq=256/1.
257	480.342994	192.168.0.110	192.168.0.97	ICMP	74	Echo (ping) reply id=0x0754, seq=256/1.
258	480.543724	PcsCompu_77:8d…	PcsCompu_97:8f…	ARP	60	Who has 192.168.0.97? Tell 192.168.0.110

No.	Time	Source	Destination	Protocc	Lengt	Info
246	447.794534	200.2.2.100	192.1.0.3	UDP	98	5001 → 5449 Len=56
247	448.832304	200.2.2.100	192.1.0.3	UDP	98	5001 → 5449 Len=56
248	449.870724	200.2.2.100	192.1.0.3	UDP	98	5001 → 5449 Len=56
249	450.322248	192.168.0.145	192.168.0.158	ICMP	74	Echo (ping) request id=0x0767, seq=256/1.
250	450.322493	192.168.0.158	192.168.0.145	ICMP	74	Echo (ping) reply id=0x0767, seq=256/1.
251	455.326059	192.168.0.145	192.168.0.158	ICMP	74	Echo (ping) request id=0x0767, seq=256/1.
252	455.326295	192.168.0.158	192.168.0.145	ICMP	74	Echo (ping) reply id=0x0767, seq=256/1.
253	458.279144	200.2.2.100	192.1.0.3	UDP	98	5001 → 8725 Len=56
254	459.316389	200.2.2.100	192.1.0.3	UDP	98	5001 → 8725 Len=56
255	460.329592	192.168.0.145	192.168.0.158	ICMP	74	Echo (ping) request id=0x0767, seq=256/1.
256	460.329813	192.168.0.158	192.168.0.145	ICMP	74	Echo (ping) reply id=0x0767, seq=256/1.
257	460.353512	200.2.2.100	192.1.0.3	UDP	98	5001 → 8725 Len=56
258	460.442544	PcsCompu_80:3d…	PcsCompu_c0:84…	ARP	60	Who has 192.168.0.145? Tell 192.168.0.158
259	460.442833	PcsCompu_c0:84…	PcsCompu_80:3d…	ARP	60	192.168.0.145 is at 08:00:27:c0:84:a4
260	461.390145	200.2.2.100	192.1.0.3	UDP	98	5001 → 8725 Len=56
261	462.427509	200.2.2.100	192.1.0.3	UDP	98	5001 → 8725 Len=56
262	465.333514	192.168.0.145	192.168.0.158	ICMP	74	Echo (ping) request id=0x0767, seq=256/1.
263	465.333732	192.168.0.158	192.168.0.145	ICMP	74	Echo (ping) reply id=0x0767, seq=256/1.
264	470.336972	192.168.0.145	192.168.0.158	ICMP	74	Echo (ping) request id=0x0767, seq=256/1.
265	470.337196	192.168.0.158	192.168.0.145	ICMP	74	Echo (ping) reply id=0x0767, seq=256/1.
266	475.340523	192.168.0.145	192.168.0.158	ICMP	74	Echo (ping) request id=0x0767, seq=256/1.

Figura 5-Captura Wireshark

Exercício 10

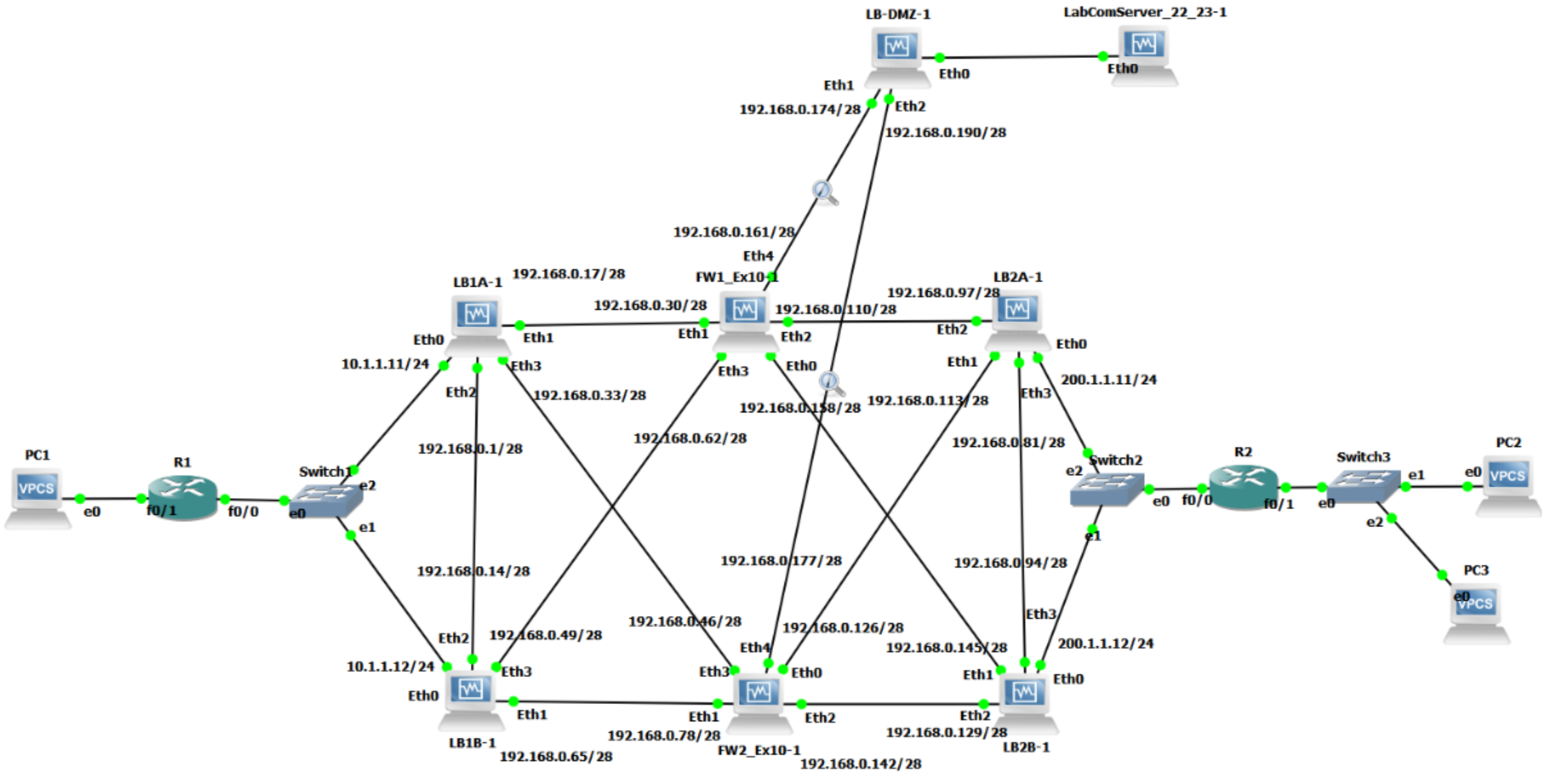


Figura 6-Rede2

Para este exercício, foram adicionadas novas configurações nas firewalls, uma nova VM, na zona DMZ, e foi adicionado também o PC3 que irá simular o endereço ip do atacante.

Em primeiro lugar, criámos uma nova zona(DMZ) e adicionámos em cada uma das firewalls uma rota estática para a rede da DMZ. Em seguida, definimos um conjunto de regras que permite os dispositivos da zona *INSIDE* comunicarem com a DMZ por TCP/UDP nos portos 22(SSH),443(HTTPS),21(FTP) e 53(DNS) e que permitem aos dispositivos da zona *OUTSIDE* comunicarem com a DMZ também por TCP/UDP, mas com acesso restrito, ou seja, neste caso, apenas o tráfego com destino nos portos de HTTPS (443) e DNS(53) é permitido.

Por fim, foram adicionadas configurações para dar drop/bloquear os pacotes com base no endereço de ip de origem(ip do atacante). De salientar que, de maneira a verificar sempre em primeiro lugar se o ip de origem é um ip identificado como sendo um atacante(dentro do range 200.2.2.20 – 200.2.2.30), definimos um menor sequence number para esta regra(rule 5).

Na figura abaixo estão presentes as configurações de uma das firewalls(FW1) e do Load Balancer que liga as firewalls à DMZ.

```
---------LB-DMZ-----------
configure
set load-balancing wan interface-health eth1 nexthop 192.168.0.161
set load-balancing wan interface-health eth2 nexthop 192.168.0.177
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
commit
save
```

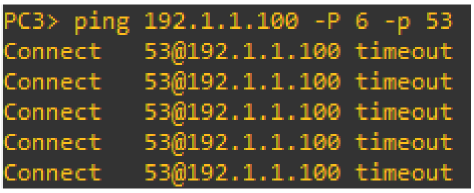
Figura 7- Configurações Load Balancer

```
----------FW1----------
configure
set zone-policy zone DMZ description "DMZ Zone"
set zone-policy zone DMZ interface eth4
commit
save

set protocols static route 192.1.1.0/24 next-hop 192.168.0.174
commit
save

set firewall name FROM-INSIDE-TO-DMZ rule 10 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 10 protocol udp
set firewall name FROM-INSIDE-TO-DMZ rule 10 destination port 22,443,21,53


set firewall name FROM-OUTSIDE-TO-DMZ rule 10 action accept
set firewall name FROM-OUTSIDE-TO-DMZ rule 10 destination port https,domain
set firewall name FROM-OUTSIDE-TO-DMZ rule 10 protocol udp

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-OUTSIDE rule 10 action accept
set firewall name FROM-DMZ-TO-OUTSIDE rule 10 state established enable

set firewall name FROM-INSIDE-TO-DMZ rule 11 action accept
set firewall name FROM-INSIDE-TO-DMZ rule 11 protocol tcp
set firewall name FROM-INSIDE-TO-DMZ rule 11 destination port 22,443,21,53

set firewall name FROM-OUTSIDE-TO-DMZ rule 11 action accept
set firewall name FROM-OUTSIDE-TO-DMZ rule 11 destination port https,domain
set firewall name FROM-OUTSIDE-TO-DMZ rule 11 protocol tcp


set firewall name FROM-OUTSIDE-TO-DMZ rule 5 action drop
set firewall group address-group attackers address '200.2.2.20-200.2.2.30'
set firewall name FROM-OUTSIDE-TO-DMZ rule 5 source group address-group attackers


set zone-policy zone DMZ from OUTSIDE firewall name FROM-OUTSIDE-TO-DMZ
set zone-policy zone DMZ from INSIDE firewall name FROM-INSIDE-TO-DMZ
set zone-policy zone INSIDE from DMZ firewall name FROM-DMZ-TO-INSIDE
set zone-policy zone OUTSIDE from DMZ firewall name FROM-DMZ-TO-OUTSIDE
commit
save
```

Figura 8-Configurações FW1

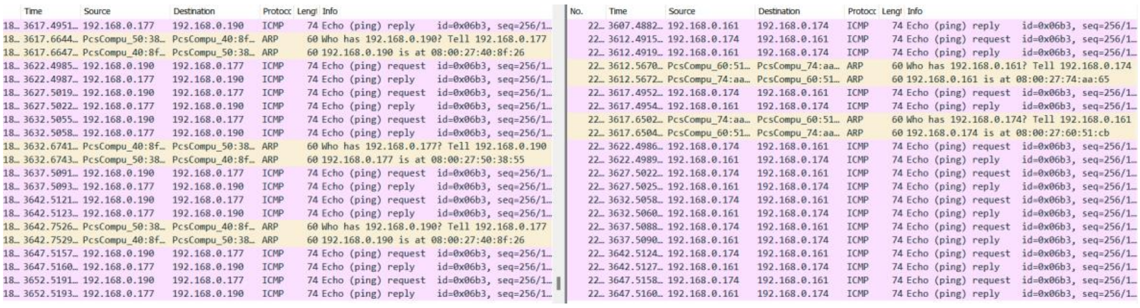
Resultados

Como era suposto, o ping do atacante para a DMZ foi bloqueado pela firewall, porque o ip do atacante é 200.2.2.22 e como definimos como regra na firewall que os ip´s de origem no range 200.2.2.20 – 200.2.2.30 iam ser bloqueados, então o ip do atacante é bloqueado. Nas capturas do wireshark(na figura 6, com a imagem da rede, dá para ver onde estão a ser feitas estas capturas), figura 9, dá para ver que nenhum pacote passou pela firewall.



```
PC3> ping 192.1.1.100 -P 6 -p 53
Connect   53@192.1.1.100 timeout
Connect   53@192.1.1.100 timeout
Connect   53@192.1.1.100 timeout
Connect   53@192.1.1.100 timeout
Connect   53@192.1.1.100 timeout
```

Figura 9-Ping Atacante

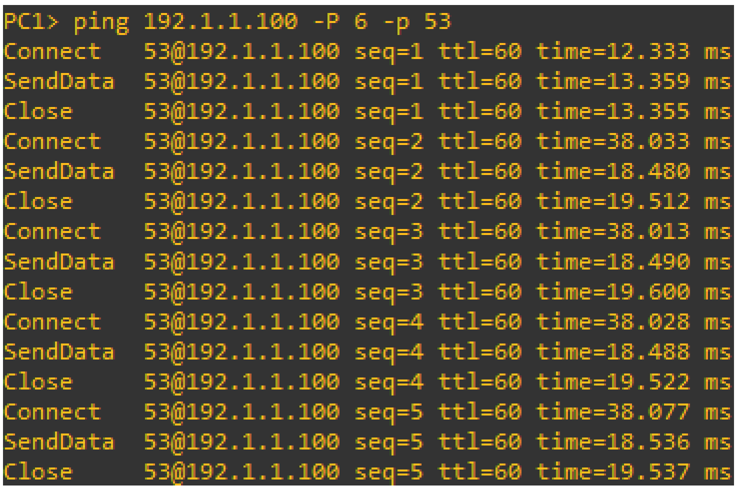


	Time	Source	Destination	Protocc	Lengt	Info
18…	3617.4951…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
18…	3617.6644…	PcsCompu_50:38…	PcsCompu_40:8f…	ARP	60	Who has 192.168.0.190? Tell 192.168.0.177
18…	3617.6647…	PcsCompu_40:8f…	PcsCompu_50:38…	ARP	60	192.168.0.190 is at 08:00:27:40:8f:26
18…	3622.4985…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
18…	3622.4987…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
18…	3627.5019…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
18…	3627.5022…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
18…	3632.5055…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
18…	3632.5058…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
18…	3632.6741…	PcsCompu_40:8f…	PcsCompu_50:38…	ARP	60	Who has 192.168.0.177? Tell 192.168.0.190
18…	3632.6743…	PcsCompu_50:38…	PcsCompu_40:8f…	ARP	60	192.168.0.177 is at 08:00:27:50:38:55
18…	3637.5091…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
18…	3637.5093…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
18…	3642.5121…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
18…	3642.5123…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
18…	3642.7526…	PcsCompu_50:38…	PcsCompu_40:8f…	ARP	60	Who has 192.168.0.190? Tell 192.168.0.177
18…	3642.7529…	PcsCompu_40:8f…	PcsCompu_50:38…	ARP	60	192.168.0.190 is at 08:00:27:40:8f:26
18…	3647.5157…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
18…	3647.5160…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
18…	3652.5191…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
18…	3652.5193…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…

No.	Time	Source	Destination	Protocc	Lengt	Info
22…	3607.4882…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	3612.4915…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	3612.4919…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	3612.5670…	PcsCompu_60:51…	PcsCompu_74:aa…	ARP	60	Who has 192.168.0.161? Tell 192.168.0.174
22…	3612.5672…	PcsCompu_74:aa…	PcsCompu_60:51…	ARP	60	192.168.0.161 is at 08:00:27:74:aa:65
22…	3617.4952…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	3617.4954…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	3617.6502…	PcsCompu_74:aa…	PcsCompu_60:51…	ARP	60	Who has 192.168.0.174? Tell 192.168.0.161
22…	3617.6504…	PcsCompu_60:51…	PcsCompu_74:aa…	ARP	60	192.168.0.174 is at 08:00:27:60:51:cb
22…	3622.4986…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	3622.4989…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	3627.5022…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	3627.5025…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	3632.5058…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	3632.5060…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	3637.5088…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	3637.5090…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	3642.5124…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	3642.5127…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	3647.5158…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	3647.5160…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…

Figura 10-Wireshark Ping Atacante

Como é possível ver nas figuras abaixo, o ping do pc na zona *INSIDE* para a DMZ foi um sucesso. Caso o ping tivesse como destino uma outra porta que não as definidas nas regras, o ping seria bloqueado.



```
PC1> ping 192.1.1.100 -P 6 -p 53
Connect   53@192.1.1.100 seq=1 ttl=60 time=12.333 ms
SendData  53@192.1.1.100 seq=1 ttl=60 time=13.359 ms
Close     53@192.1.1.100 seq=1 ttl=60 time=13.355 ms
Connect   53@192.1.1.100 seq=2 ttl=60 time=38.033 ms
SendData  53@192.1.1.100 seq=2 ttl=60 time=18.480 ms
Close     53@192.1.1.100 seq=2 ttl=60 time=19.512 ms
Connect   53@192.1.1.100 seq=3 ttl=60 time=38.013 ms
SendData  53@192.1.1.100 seq=3 ttl=60 time=18.490 ms
Close     53@192.1.1.100 seq=3 ttl=60 time=19.600 ms
Connect   53@192.1.1.100 seq=4 ttl=60 time=38.028 ms
SendData  53@192.1.1.100 seq=4 ttl=60 time=18.488 ms
Close     53@192.1.1.100 seq=4 ttl=60 time=19.522 ms
Connect   53@192.1.1.100 seq=5 ttl=60 time=38.077 ms
SendData  53@192.1.1.100 seq=5 ttl=60 time=18.536 ms
Close     53@192.1.1.100 seq=5 ttl=60 time=19.537 ms
```

Figura 11-Ping Inside

	Time	Source	Destination	Protocc	Lengt	Info
21…	4204.2019…	10.2.2.100	192.1.1.100	TCP	74	17163 → 53 [SYN] Seq=0 Win=2920 Len=0 MSS…
21…	4204.2026…	192.1.1.100	10.2.2.100	TCP	74	53 → 17163 [SYN, ACK] Seq=0 Ack=1 Win=651…
21…	4204.2225…	10.2.2.100	192.1.1.100	TCP	66	17163 → 53 [ACK] Seq=1 Ack=1 Win=5840 Len…
21…	4204.2226…	10.2.2.100	192.1.1.100	TCP	122	17163 → 53 [PSH, ACK] Seq=1 Ack=1 Win=584…
21…	4204.2230…	192.1.1.100	10.2.2.100	TCP	66	53 → 17163 [ACK] Seq=1 Ack=57 Win=65152 L…
21…	4204.2430…	10.2.2.100	192.1.1.100	TCP	66	17163 → 53 [FIN, PSH, ACK] Seq=57 Ack=1 W…
21…	4204.2436…	192.1.1.100	10.2.2.100	TCP	66	53 → 17163 [FIN, ACK] Seq=1 Ack=58 Win=65…
21…	4204.2636…	10.2.2.100	192.1.1.100	TCP	66	[TCP Keep-Alive] 17163 → 53 [ACK] Seq=57 …
21…	4204.2641…	192.1.1.100	10.2.2.100	TCP	66	[TCP Keep-Alive ACK] 53 → 17163 [ACK] Seq…
22…	4204.4716…	192.1.1.100	10.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
22…	4204.9474…	192.1.1.100	10.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
22…	4205.2607…	10.2.2.100	192.1.1.100	TCP	74	[TCP Port numbers reused] 17163 → 53 [SYN…
22…	4205.2615…	192.1.1.100	10.2.2.100	TCP	66	53 → 17163 [ACK] Seq=1 Ack=2871193468 Win…
22…	4205.2812…	10.2.2.100	192.1.1.100	TCP	66	17163 → 53 [RST, ACK] Seq=2871193468 Ack=…
22…	4207.9153…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	4207.9156…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
22…	4208.6220…	10.2.2.100	192.1.1.100	TCP	74	[TCP Port numbers reused] 17163 → 53 [SYN…
22…	4208.6227…	192.1.1.100	10.2.2.100	TCP	66	[TCP ACKed unseen segment] 53 → 17163 [AC…
22…	4208.6425…	10.2.2.100	192.1.1.100	TCP	66	17163 → 53 [RST, ACK] Seq=735693199 Ack=1…
22…	4212.9185…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
22…	4212.9187…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…

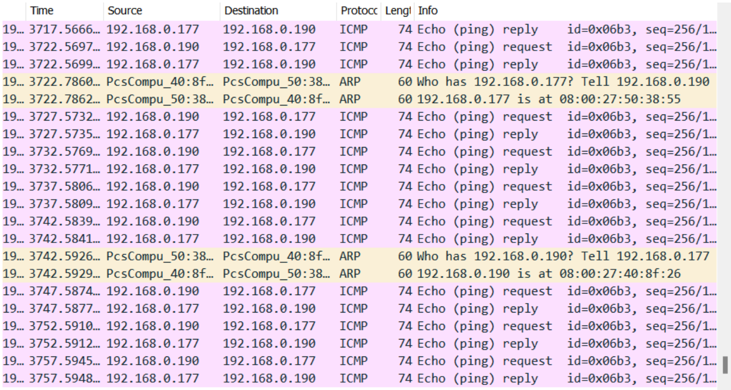
No.	Time	Source	Destination	Protocc	Lengt	Info
28…	4202.6214…	10.2.2.100	192.1.1.100	TCP	66	[TCP Keep-Alive] 17163 → 53 [ACK] Seq=57 …
28…	4202.6219…	192.1.1.100	10.2.2.100	TCP	66	[TCP Keep-Alive ACK] 53 → 17163 [ACK] Seq…
28…	4202.8279…	192.1.1.100	10.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
28…	4202.9120…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
28…	4202.9122…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
28…	4203.3044…	192.1.1.100	10.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
28…	4203.6392…	10.2.2.100	192.1.1.100	TCP	74	[TCP Port numbers reused] 17163 → 53 [SYN…
28…	4203.6399…	192.1.1.100	10.2.2.100	TCP	74	53 → 17163 [SYN, ACK] Seq=0 Ack=1 Win=651…
28…	4203.6598…	10.2.2.100	192.1.1.100	TCP	66	17163 → 53 [ACK] Seq=1 Ack=1 Win=5840 Len…
28…	4203.6803…	10.2.2.100	192.1.1.100	TCP	122	17163 → 53 [PSH, ACK] Seq=1 Ack=1 Win=584…
28…	4203.6809…	192.1.1.100	10.2.2.100	TCP	66	53 → 17163 [ACK] Seq=1 Ack=57 Win=65152 L…
28…	4203.7215…	10.2.2.100	192.1.1.100	TCP	66	17163 → 53 [FIN, PSH, ACK] Seq=57 Ack=1 W…
28…	4203.7222…	192.1.1.100	10.2.2.100	TCP	66	53 → 17163 [FIN, ACK] Seq=1 Ack=58 Win=65…
28…	4203.7421…	10.2.2.100	192.1.1.100	TCP	66	[TCP Keep-Alive] 17163 → 53 [ACK] Seq=57 …
28…	4203.7427…	192.1.1.100	10.2.2.100	TCP	66	[TCP Keep-Alive ACK] 53 → 17163 [ACK] Seq…
28…	4203.9479…	192.1.1.100	10.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
28…	4204.4239…	192.1.1.100	10.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
28…	4205.3520…	192.1.1.100	10.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
28…	4207.1762…	192.1.1.100	10.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
28…	4207.9152…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
28…	4207.9154…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…

Figura 12-Wireshark Ping Inside

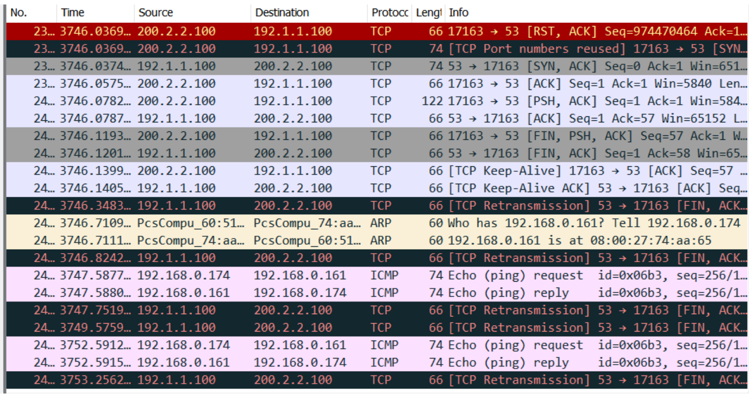
Por fim, testámos o ping do *OUTSIDE* para a DMZ e verificamos que também teve resposta, tal como é possível ver nas figuras abaixo.

```
PC2> ping 192.1.1.100 -P 6 -p 53
Connect   53@192.1.1.100 seq=1 ttl=60 time=20.552 ms
SendData  53@192.1.1.100 seq=1 ttl=60 time=18.543 ms
Close     53@192.1.1.100 seq=1 ttl=60 time=18.507 ms
Connect   53@192.1.1.100 seq=2 ttl=60 time=38.047 ms
SendData  53@192.1.1.100 seq=2 ttl=60 time=18.518 ms
Close     53@192.1.1.100 seq=2 ttl=60 time=19.559 ms
Connect   53@192.1.1.100 seq=3 ttl=60 time=38.028 ms
SendData  53@192.1.1.100 seq=3 ttl=60 time=18.522 ms
Close     53@192.1.1.100 seq=3 ttl=60 time=19.553 ms
Connect   53@192.1.1.100 seq=4 ttl=60 time=38.017 ms
SendData  53@192.1.1.100 seq=4 ttl=60 time=18.495 ms
Close     53@192.1.1.100 seq=4 ttl=60 time=19.528 ms
Connect   53@192.1.1.100 seq=5 ttl=60 time=36.997 ms
SendData  53@192.1.1.100 seq=5 ttl=60 time=19.526 ms
Close     53@192.1.1.100 seq=5 ttl=60 time=18.524 ms
```

Figura 13-Ping Outside



	Time	Source	Destination	Protocc	Lengt	Info
19…	3717.5666…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
19…	3722.5697…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
19…	3722.5699…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
19…	3722.7860…	PcsCompu_40:8f…	PcsCompu_50:38…	ARP	60	Who has 192.168.0.177? Tell 192.168.0.190
19…	3722.7862…	PcsCompu_50:38…	PcsCompu_40:8f…	ARP	60	192.168.0.177 is at 08:00:27:50:38:55
19…	3727.5732…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
19…	3727.5735…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
19…	3732.5769…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
19…	3732.5771…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
19…	3737.5806…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
19…	3737.5809…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
19…	3742.5839…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
19…	3742.5841…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
19…	3742.5926…	PcsCompu_50:38…	PcsCompu_40:8f…	ARP	60	Who has 192.168.0.190? Tell 192.168.0.177
19…	3742.5929…	PcsCompu_40:8f…	PcsCompu_50:38…	ARP	60	192.168.0.190 is at 08:00:27:40:8f:26
19…	3747.5874…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
19…	3747.5877…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
19…	3752.5910…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
19…	3752.5912…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
19…	3757.5945…	192.168.0.190	192.168.0.177	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
19…	3757.5948…	192.168.0.177	192.168.0.190	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…



No.	Time	Source	Destination	Protocc	Lengt	Info
23…	3746.0369…	200.2.2.100	192.1.1.100	TCP	66	17163 → 53 [RST, ACK] Seq=974470464 Ack=1…
23…	3746.0369…	200.2.2.100	192.1.1.100	TCP	74	[TCP Port numbers reused] 17163 → 53 [SYN…
23…	3746.0374…	192.1.1.100	200.2.2.100	TCP	74	53 → 17163 [SYN, ACK] Seq=0 Ack=1 Win=651…
23…	3746.0575…	200.2.2.100	192.1.1.100	TCP	66	17163 → 53 [ACK] Seq=1 Ack=1 Win=5840 Len…
24…	3746.0782…	200.2.2.100	192.1.1.100	TCP	122	17163 → 53 [PSH, ACK] Seq=1 Ack=1 Win=584…
24…	3746.0787…	192.1.1.100	200.2.2.100	TCP	66	53 → 17163 [ACK] Seq=1 Ack=57 Win=65152 L…
24…	3746.1193…	200.2.2.100	192.1.1.100	TCP	66	17163 → 53 [FIN, PSH, ACK] Seq=57 Ack=1 W…
24…	3746.1201…	192.1.1.100	200.2.2.100	TCP	66	53 → 17163 [FIN, ACK] Seq=1 Ack=58 Win=65…
24…	3746.1399…	200.2.2.100	192.1.1.100	TCP	66	[TCP Keep-Alive] 17163 → 53 [ACK] Seq=57 …
24…	3746.1405…	192.1.1.100	200.2.2.100	TCP	66	[TCP Keep-Alive ACK] 53 → 17163 [ACK] Seq…
24…	3746.3483…	192.1.1.100	200.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
24…	3746.7109…	PcsCompu_60:51…	PcsCompu_74:aa…	ARP	60	Who has 192.168.0.161? Tell 192.168.0.174
24…	3746.7111…	PcsCompu_74:aa…	PcsCompu_60:51…	ARP	60	192.168.0.161 is at 08:00:27:74:aa:65
24…	3746.8242…	192.1.1.100	200.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
24…	3747.5877…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
24…	3747.5880…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
24…	3747.7519…	192.1.1.100	200.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
24…	3749.5759…	192.1.1.100	200.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…
24…	3752.5912…	192.168.0.174	192.168.0.161	ICMP	74	Echo (ping) request id=0x06b3, seq=256/1…
24…	3752.5915…	192.168.0.161	192.168.0.174	ICMP	74	Echo (ping) reply id=0x06b3, seq=256/1…
24…	3753.2562…	192.1.1.100	200.2.2.100	TCP	66	[TCP Retransmission] 53 → 17163 [FIN, ACK…

Figura 14- Wireshark Ping Outside

Script

Este script tem como objetivo criar automaticamente as regras de bloqueio da firewall após a identificação do endereço de ip dos atacantes.

Foi definido um range de endereços para simular a lista com ip´s dos atacantes. Em seguida, através do comando da linha do *tcpdump …* capturamos os pacotes na interface enp0s3 filtrando os pacotes que têm o endereço ip no range definido pelas variáveis *$ip_start e $ip_end* e parar após capturar 1000 pacotes. Ainda na mesma linha, o *while read attacker_ip* vai iterar sobre cada linha e vai atribuir o valor à variável *attacker_ip.*

Posto isto, se pacotes forem capturados nesta interface através do comando *tcpdump* significa que houve pacotes cujo ip origem estava na lista dos endereços ip´s dos atacantes e iniciamos uma conexão remota via ssh para as duas firewall para bloquear pacotes com base no endereço ip de origem(attacker_ip).

Infelizmente tentámos testar este script, mas não conseguimos chegar ao resultado pretendido, mas pelo menos fica uma ideia daquilo que pretendíamos fazer no script.

```
#!/bin/bash

# Set the firewall IP address
firewall1_ip_address="firewall1_ip_address"
firewall2_ip_address="firewall2_ip_address"

# Set the IP range to block
ip_start="200.2.2.20"
ip_end="200.2.2.30"


echo "Searching for DDOS attacks"
# Start capturing network traffic with tcpdump and filter it based on source IP address
tcpdump -n -i enp0s3 src net $ip_start/$ip_end -c 1000 | grep -oE "\b([0-9]{1,3}.){3}[0-9]{1,3}\b" | sort -u | while read attacker_ip;
do
    # Connect to the firewall1 via SSH
    ssh firewall1_ip_address << EOF
        configure
      set firewall name BLOCK rule 10 action drop
          set firewall name BLOCK rule 10 source address $attacker_ip
        commit
        save
        exit
EOF
    # Connect to the firewall2 via SSH
    ssh firewall2_ip_address << EOF
        configure
      set firewall name BLOCK rule 10 action drop
          set firewall name BLOCK rule 10 source address $attacker_ip
        commit
        save
        exit
EOF

    # Print a message to confirm that the blocking rule has been added
    echo "Blocking rule added to the firewalls to block traffic from IP address $attacker_ip"
done
```

Figura 15- Script

Conclusão

Concluindo, achamos que conseguimos atingir todos os objetivos deste projeto com sucesso. Salientando que, apesar de não termos mostrado os resultados dos pings em todas as portas das regras, os mesmos tiveram, tal como os outros, sucesso.