Trabalho Laboratorial 3

Planeamento e Gestão de Redes

Endereçamento IP e Serviço DNS

Diogo Remião & Miguel Pinheiro

Abril 2021

Faculdade de Engenharia da Universidade do Porto
TEC

Conteúdos

1 Introdução

Este trabalho tem como objetivos a compreensão de requisitos de endereçamento IP numa rede empresarial e configuração de servidores DNS locais. Para tal, foi desenvolvido um projeto de endereçamento tendo em conta os diferentes sub-sistemas da rede empresarial, nomeadamente a criação de uma intranet local, sem acesso à internet, e uma DMZ, que funciona como a fronteira para o exterior da empresa, com acesso à internet. Todas estes sub-sistemas terão a sua própria rede local, pelo que será necessário configurar diferentes Vlans para este efeito.

Depois do plano endereçamento, irá ser abordada a configuração de servidores DNS dentro da empresa, com o objetivo de resolver domínios locais. Estes endereços apenas poderão ser visíveis nas redes locais permitidas, pelo que será utilizada uma bordagem de *Split DNS* para filtrar os pedidos.

Por fim, iremos configurar a rede de forma a que apenas a DMZ tenha acesso à internet, através do protocolo NAT.

2 Endereçamento

2.1 Atribuição de endereços

De acordo com a estrutura da rede e as suas dependências, foi elaborado o seguinte endereçamento da rede:

Endereço IP	Network	Broadcast	DNS	Gateway
Sede	192.168.0.0/23	192.168.1.255	-	192.168.1.254
Rede de Servidores	192.168.2.224/29	192.168.2.231	192.168.2.225	192.168.2.230
Armazém	192.168.2.192/27	192.168.2.223	192.168.2.193	192.168.2.222
Loja1 - Vlan1	192.168.2.0/27	192.168.2.31	192.168.2.1	192.168.2.30
Loja1 - Vlan2	192.168.2.32/27	192.168.2.63	192.168.2.33	192.168.2.62
Loja2 - Vlan1	192.168.2.64/27	192.168.2.95	192.168.2.65	192.168.2.94
Loja2 - Vlan2	192.168.2.96/27	192.168.2.127	192.168.2.97	192.168.2.126
Loja3 - Vlan1	192.168.2.128/27	192.168.2.159	192.168.2.129	192.168.2.158
Loja3 - Vlan2	192.168.2.160/27	192.168.2.191	192.168.2.161	192.168.2.190
DMZ	20.49.51.160/28	20.49.51.175	20.49.51.161	20.49.51.174

Table 2.1: Endereços dos elementos da rede

2.2 Configuração na bancada

Para simular a tipologia da rede empresarial, foi criada a seguinte estrutura de rede na bancada:

Host	Eth1	Eth2
Tux12	Desativada	DNS Cache Armazém
Tux13	DNS Rede de Servidores	DNS DMZ
Tux14	DNS Loja 1 - Vlan1	DNS Loja 1 - Vlan2

Table 2.2: Estrutura de rede nos hosts

Para se obter esta tipologia, as respetivas interfaces foram configuradas deste modo:

```
eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.193  netmask 255.255.255.224  broadcast 192.168.2.223
        inet6 fe80::2c0:dfff:fe08:8125  prefixlen 64  scopeid 0x20<link>
        ether 00:c0:df:08:81:25  txqueuelen 1000  (Ethernet)
        RX packets 3052  bytes 578186 (564.6 KiB)
        RX errors 0  dropped 2998  overruns 0  frame 0
        TX packets 211  bytes 21964 (21.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 2.1: Tux12 Interfaces

```
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.225  netmask 255.255.255.248  broadcast 192.168.2.231
        inet6 fe80::250:fcff:feed:bb37  prefixlen 64  scopeid 0x20<link>
        ether 00:50:fc:ed:bb:37  txqueuelen 1000  (Ethernet)
        RX packets 17827  bytes 2845382 (2.7 MiB)
        RX errors 0  dropped 12303  overruns 0  frame 0
        TX packets 416  bytes 38149 (37.2 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 20.49.51.161  netmask 255.255.255.240  broadcast 20.49.51.175
        inet6 fe80::201:2ff:fea0:fa2e  prefixlen 64  scopeid 0x20<link>
        ether 00:01:02:a0:fa:2e  txqueuelen 1000  (Ethernet)
        RX packets 33823  bytes 2466259 (2.3 MiB)
        RX errors 0  dropped 33769  overruns 0  frame 0
        TX packets 286  bytes 26694 (26.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 21  base 0x5100
```

Figure 2.2: Tux13 Interfaces

```
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.1  netmask 255.255.255.224  broadcast 192.168.2.31
        inet6 fe80::2c0:dfff:fe04:2099  prefixlen 64  scopeid 0x20<link>
        ether 00:c0:df:04:20:99  txqueuelen 1000  (Ethernet)
        RX packets 134423  bytes 9686869 (9.2 MiB)
        RX errors 0  dropped 133989  overruns 0  frame 0
        TX packets 783  bytes 84156 (82.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.33  netmask 255.255.255.224  broadcast 192.168.2.63
        inet6 fe80::201:2ff:fe21:8305  prefixlen 64  scopeid 0x20<link>
        ether 00:01:02:21:83:05  txqueuelen 1000  (Ethernet)
        RX packets 34778  bytes 2534194 (2.4 MiB)
        RX errors 0  dropped 34728  overruns 0  frame 0
        TX packets 198  bytes 23370 (22.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 21  base 0xd100
```

Figure 2.3: Tux14 Interfaces

2.3 Switch / Router

A estrutura de rede apresentada requere que os diferentes sistemas estejam em Vlans diferentes, de forma a que o seu tráfego interno seja separado. Para tal, várias Vlans foram configuradas no Switch de bancada, atribuindo as portas a que as interfaces dos Tux's estavam ligadas às Vlans respetivas [1]:

```
VLAN Name                             Status    Ports
---- -------------------------------- --------- -------------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4, Fa0/5
                                                Fa0/6, Fa0/7, Fa0/8, Fa0/9, Fa0/10
                                                Fa0/11, Fa0/12, Fa0/13, Fa0/14, Fa0/16
                                                Fa0/17, Fa0/20, Fa0/21, Fa0/24, Gi0/1
2    Sede                             active    Fa0/15
3    Armazem                          active    Fa0/19
4    Loja1                            active    Fa0/23
5    Loja2                            active    Fa0/22
6    Dmz                              active    Fa0/18
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup

VLAN Type  SAID       MTU   Parent RingNo BridgeNo Stp  BrdgMode Trans1 Trans2
---- ----- ---------- ----- ------ ------ -------- ---- -------- ------ ------
1    enet  100001     1500  -      -      -        -    -        0      0
2    enet  100002     1500  -      -      -        -    -        0      0
3    enet  100003     1500  -      -      -        -    -        0      0
4    enet  100004     1500  -      -      -        -    -        0      0
5    enet  100005     1500  -      -      -        -    -        0      0
6    enet  100006     1500  -      -      -        -    -        0      0
```

Figure 2.4: Switch Vlans

O switch não faz *forward* de pacotes entre diferentes Vlans. Desse modo, se queremos que as vlans possam comunicar entre si, temos que fazer uso de um dispositivo que suporte *routing* Layer 3. Neste caso, esse dispositivo é router de bancada, o que nos obriga a usar a abordagem **ROAS** (Router-on-a-stick).

Apenas existe uma interface que liga o Switch ao Router : GigabitEthernet2. A configuração ROAS permite que esta interface seja dividida em sub-interfaces, cada uma delas associada a uma VLAN [2]. Estas sub-interfaces vão funcionar como as Gateways das diferentes Vlans no router.

O primeiro passo é criar um **Trunk** no Switch na interface que liga ao Router, que junte todas as vlans. Esta trunk permite que as vlans sejam discriminadas à chegada ao router através do encapsulamento *dot1Q*.

```
interface GigabitEthernet0/2
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 1-6
 switchport mode trunk
```

Figure 2.5: Switch Trunk

Após a criação do trunk, conseguimos no router separar as Vlans que estão nesse trunk e atribuir a cada uma a sua Gateway.

```
interface GigabitEthernet0/0
 description switch-trunk
 ip address 172.16.1.19 255.255.255.0
 duplex auto
 speed auto
!
interface GigabitEthernet0/0.2
 encapsulation dot1Q 2
 ip address 192.168.2.230 255.255.255.248
!
interface GigabitEthernet0/0.3
 encapsulation dot1Q 3
 ip address 192.168.2.222 255.255.255.224
!
interface GigabitEthernet0/0.4
 encapsulation dot1Q 4
 ip address 192.168.2.30 255.255.255.224
!
interface GigabitEthernet0/0.5
 encapsulation dot1Q 5
 ip address 192.168.2.62 255.255.255.224
!
interface GigabitEthernet0/0.6
 encapsulation dot1Q 6
 ip address 20.49.51.174 255.255.255.240
!
```

Figure 2.6: Router Interfaces

Ao configurar o router, atribuímos a cada Vlan a sua *Gateway*, assim como a sua *Netmask* correspondente. Deste modo, virtualizamos uma ligação direta individual de cada Vlan ao router.

Para testar a configuração podemos fazer um *traceroute* entre duas Vlans distintas:

```
root@tux13:~# traceroute -i eth1 192.168.2.193
traceroute to 192.168.2.193 (192.168.2.193), 30 hops max, 60 byte packets
 1  192.168.2.230 (192.168.2.230)  0.472 ms  0.544 ms  0.573 ms
 2  192.168.2.193 (192.168.2.193)  0.402 ms  0.397 ms  0.380 ms
root@tux13:~#
```

Figure 2.7: Traceroute da Rede servidores para o Armazém

Podemos observar que a Vlan da rede de servidores faz uso da sua gateway para

aceder ao DNS da Vlan do Armazém.

3 Configuração DNS / Bind9

3.1 Considerações

Para ser possível comunicação entre os diferentes sub-sistemas da rede através dos respetivos domínios sem estarmos a depender de IPs, é necessário criar um servidor DNS. O software escolhido foi o **BIND9**, que permite também a configuração do servidor para cache DNS.

Na estrutura de rede que estamos a analisar, temos uma rede local segura e excluída do exterior, e uma DMZ, ligada ao exterior. Idealmente, um servidor DNS seria configurado em cada uma das zonas. No entanto, devido a limitações de hardware, a DMZ está na mesmo *host* que a rede de computadores, pelo que o servidor DNS é partilhado. Desse modo, é preciso configurar uma **Split DNS**. Esta configuração permite a distinção dos endereços que fazem as *queries*. É assim possível fornecer uma resposta diferente dependendo do *host* que está a fazer a *query*.

3.2 Rede de Servidores / DMZ

O servidor DNS na rede de servidores tem como objetivo devolver o endereço para os serviços de:

- Nameserver - ns.qquma.pt
- Webserver - www.qquma.pt
- Mailserver - mail.qquma.pt

No entanto, o endereço a ser devolvido depende da origem do pedido. Se o pedido for proveniente da rede interna (192.168.0.0/21), os endereços a devolver correspondem aos serviços alojados na rede de servidores (192.168.2.22x). Se o pedido for externo (incluíndo da DMZ), os endereços a devolver correspondem aos serviços alojados na DMZ (20.49.51.16x).

Além disso, o servidor DNS só pode resolver pedidos para domínios da loja e do armazém a pedidos internos na rede, bloqueando pedidos externos.

Existe mais que uma forma de fazer esta separação no DNS. A primeira, que é o exemplo fornecido pela documentação do Bind para uma *Split-DNS* [3], faz uso das funções `allow-query{}` que especifica os endereços que permite *queries* e `allow-forward{}` que especifica os endereços que permite encaminhamento de *queries* ao *slave*.

O segundo método, que foi o escolhido, prende-se com a criação de *views*, que é um conjunto de zonas que seguem as mesmas configurações gerais [4]. Neste caso, definimos uma *view* para pedidos internos, que através da função `match-clients{}` permite especificar os endereços que entram nessa *view*. Estas *views* funcionam por exclusão, isto é, o pedido tenta entrar na *views* por ordem e só entra na *view* geral se não conseguir entrar na anterior.

Em ambos os casos, definimos ficheiros *database* diferentes para as zonas internas e externas.

```
acl "company" {
        192.168.0.0/21;
 };

view "internal" { // internal view of the companies zons
        match-clients { "company"; };

        allow-recursion { "company"; };
        zone "qquma.pt"{
                type master;
                file "/var/named/db.qquma"; //internal db
        };


        zone "ns.loja1.qquma.pt"{
                type slave;
                masters{ 192.168.2.1; };
        };

};

view "external" { //external view of the companies zones to the internel
        match-clients { any; }; //obvious

        zone "qquma.pt" {
                type master;
                file "/var/named/db.qquma.external"; //external db
        };

};
// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
```

Figure 3.1: Ficheiro named.conf.local Tux13

Definimos o *range* de IPs 192.168.0.0/21 como pedidos locais e todos os outros como externos.

No ficheiro **db.qquma**, definimos os *DNS Records* para resolver pedidos internos para os diferentes serviços.

```
$TTL    604800
@       IN      SOA     ns.qquma.pt. mail.qquma.pt. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.qquma.pt.
@       IN      A       192.168.2.225
ns      IN      A       192.168.2.225
www     IN      A       192.168.2.226
mail    IN      A       192.168.2.227
armazem IN      A       192.168.2.193
```

Figure 3.2: Ficheiro db.qquma no Tux13

No ficheiro **db.qquma.external**, definimos os *DNS Records* para resolver pedidos internos para os diferentes serviços.

```
$TTL    604800
@       IN      SOA     ns.qquma.pt. mail.qquma.pt. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.qquma.pt.
@       IN      A       20.49.51.161
ns      IN      A       20.49.51.161
www     IN      A       20.49.51.163
mail    IN      A       20.49.51.162
```

Figure 3.3: Ficheiro db.qquma.external no Tux13

Por fim, é preciso especificar no ficheiro `/etc/resolv.conf` o servidor DNS que o sistema tem que usar, que neste caso é a si próprio.

3.3 Armazém

O servidor DNS do armazém tem características diferentes. Primeiramente, este servidor tem que suportar *Caching*, pelo que essa funcionalidade tem que ser ativada.

Em segundo lugar, o servidor DNS não resolve nenhuma *query*, nem mesmo para si próprio. Ele tem que reencaminhar os pedidos DNS para a o DNS da rede de servidores, neste caso na *view* interna. Deste modo, não é preciso configurar um ficheiro *database*,

mas é necessário definir o servidor DNS na rede de servidores como *forwarder*. Isto vai reencaminhar os pedidos que o servidor local não sabe resolver, que neste caso é todos.

```
options {

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        directory "/var/cache/bind";

        forwarders {
                192.168.2.225;
        };

        recursion yes;

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
        dnssec-enable yes;
        dnssec-validation yes;
        auth-nxdomain no;
        listen-on-v6 { any; };
};
```

Figure 3.4: Ficheiro named.conf.options no Tux12

3.4 Loja

O servidor DNS da loja tem que conseguir resolver o seu próprio endereço. Deste modo, este servidor DNS é o *slave* do servidor DNS da rede de servidores Sempre que um *host* pede para resolver o *domain* da loja, o servidor DNS da rede de servidores vai reencaminhar o pedido para este servidor, que o vai resolver. Note-se que este pedido apenas é redirecionado na *view* interna, isto é, para pedidos locais. Assim garante-se que o domain não é resolvido para IPs externos..

```
//
// Do any local configuration here
//

zone "loja1.qquma.pt"{
        type master;
        file "/var/named/db.loja1.qquma";
};

zone "qquma.pt"{
        type slave;
        masters{ 192.168.2.225; };
};

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
```

Figure 3.5: Ficheiro named.conf.local no Tux14

É preciso indicar no ficheiro de configurações qual é a *database* que o DNS tem que aceder para resolver os endereços.

```
$TTL    604800
@       IN      SOA     loja1.qquma.pt. mail.qquma.pt. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      loja1.qquma.pt.
@       IN      A       192.168.2.1
proxy   IN      A       192.168.2.2
```

Figure 3.6: Ficheiro db.loja1.qquma no Tux14

No ficheiro verb|db.loja1.qquma| definimos os DNS Records quer para o próprio endereço do *Nameserver*, quer para o *Proxy*.

4 Configuração do NAT

NAT (Network Address Translation) é um protocolo de atribuição de endereços IP público não atribuídos para acesso à Internet [5]. Permite a conservação do endereços IP, de modo a garantir que não há endereços IP reservados a não ser utilizados, dado que há um número finito de endereços disponíveis.

Na estrutura de rede considerada, apenas a DMZ tem acesso à internet. Deste modo, apenas temos que configurar o NAT para as interfaces usadas pela DMZ.

```
interface GigabitEthernet0/0
 description switch-trunk
 ip address 172.16.1.19 255.255.255.0
 ip nat outside
 ip virtual-reassembly
 duplex auto
 speed auto
!
interface GigabitEthernet0/0.2
 encapsulation dot1Q 2
 ip address 192.168.2.230 255.255.255.248
!
interface GigabitEthernet0/0.3
 encapsulation dot1Q 3
 ip address 192.168.2.222 255.255.255.224
!
interface GigabitEthernet0/0.4
 encapsulation dot1Q 4
 ip address 192.168.2.30 255.255.255.224
!
interface GigabitEthernet0/0.5
 encapsulation dot1Q 5
 ip address 192.168.2.62 255.255.255.224
!
interface GigabitEthernet0/0.6
 encapsulation dot1Q 6
 ip address 20.49.51.174 255.255.255.240
 ip nat inside
 ip virtual-reassembly
!
```

Figure 4.1: Configuração do NAT nas interfaces GE0/0 e GE0/0.6

Para configurar o NAT, é sempre preciso definir uma interface *NAT inside* e outra *NAT outside*.

A interface **NAT Inside** corresponde à origem do tráfego interno que tem como destino a internet. Neste caso, o tráfego provém apenas da DMZ, pelo que apenas a sua interface foi configurada (GE0/0.6)

A interface **NAT Outside** corresponde à interface que tem ligação ao exterior, isto é, à internet. Neste caso, a interface é que se tem que definir é aquela que está ligada ao *Firetux* (GE0/0).

Para testar se o NAT foi bem configurado, fazemos um *ping* a um endereço externo, como por exemplo `google.com`.

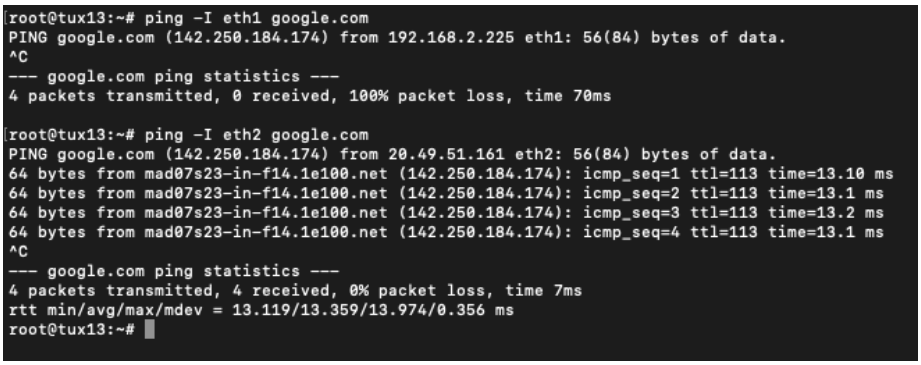

Figure 4.2: Ping nas interfaces eth1 e eth2

Como se observa, a interface **eth1** que corresponde à rede de servidores, que não tem acesso à internet. Com a interface **eth2** que corresponde à DMZ, o *ping* é bem sucedido. Nas restantes interfaces / sistemas, o *ping* também não é bem sucedido como seria de esperar.

5 Teste da configuração do DNS

Para testar as configurações dos DNS, foi ativada a opção de logging no servidor DNS da rede de servidores para podermos analisar os pedidos DNS recebidos. Os logs são escritos no ficheiro `/var/log/query.log`.

```
logging{
        channel query_logging{
                file "/var/log/query.log" versions 3 size 10m;
                severity debug 3;
                print-time yes;
                print-severity yes;
                print-category yes;
        };

        category queries {
                query_logging;
        };
};
```

Figure 5.1: Configuração do Logging

É feito uso do comando `dig` para fazer as queries DNS. A razão para a utilização deste comando ao invés de `nslookup` deve-se ao facto de ser possível discriminar a interface desejada.

5.1 Split DNS

O **Split DNS** vai separar os pedidos conforme o IP de origem. Desse modo é de esperar:

- Pedidos locais: Endereços da rede de servidores
- Pedidos Externos: Endereços da DMZ

```
root@tux13:/# dig -b 192.168.2.225 www.qquma.pt

; <<>> DiG 9.11.5-P4-5.1+deb10u3-Debian <<>> -b 192.168.2.225 www.qquma.pt
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20947
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: be2c7b82989552a6910503576085de9988fa44541bc746f0 (good)
;; QUESTION SECTION:
;www.qquma.pt.                  IN      A

;; ANSWER SECTION:
www.qquma.pt.           604800  IN      A       192.168.2.226

;; AUTHORITY SECTION:
qquma.pt.               604800  IN      NS      ns.qquma.pt.

;; ADDITIONAL SECTION:
ns.qquma.pt.            604800  IN      A       192.168.2.225

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Apr 25 22:26:49 WEST 2021
;; MSG SIZE  rcvd: 118
```

Figure 5.2: Query de um endereço local

```
25-Apr-2021 22:26:49.211 queries: info: client @0x7f07880bbf40 192.168.2.225#50803 (www.qquma.pt): view internal
```

Figure 5.3: Log da query de endereço local

O endereço devolvido pela *Query* é `192.168.2.226`, que corresponde ao Webserver da rede de servidores. No log é possível verificar que a *view* do pedido é a interna, confirmado que reconhece o endereço local.

```
root@tux13:/# dig -b 172.16.1.13 www.qquma.pt

; <<>> DiG 9.11.5-P4-5.1+deb10u3-Debian <<>> -b 172.16.1.13 www.qquma.pt
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24717
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
; COOKIE: 513824b93e43eee27770b41f6085df49b4106cbe0e8600aa (good)
;; QUESTION SECTION:
;www.qquma.pt.                  IN      A

;; ANSWER SECTION:
www.qquma.pt.           604800  IN      A       20.49.51.163

;; AUTHORITY SECTION:
qquma.pt.               604800  IN      NS      ns.qquma.pt.

;; ADDITIONAL SECTION:
ns.qquma.pt.            604800  IN      A       20.49.51.161

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Sun Apr 25 22:29:45 WEST 2021
;; MSG SIZE  rcvd: 118
```

Figure 5.4: Query de um endereço da DMZ

```
25-Apr-2021 22:29:45.031 queries: info: client @0x7f07880e47a0 172.16.1.13#37644 (www.qquma.pt): view external
```

Figure 5.5: Log da query de endereço da DMZ

Neste caso, o endereço devolvido pela *Query* é `20.49.51.163`, que corresponde ao Webserver da DMZ. Analogamente, no log é possível verificar que a *view* do pedido é a externa, confirmado que filtra o pedido como externo.

É de notar que os pedidos tem um *query time* de 0 ms, o que confirma que os pedidos são resolvidos localmente.

5.2 Loja

Os pedidos com origem na loja são tratados como pedidos locais pelo que devem ser devolvidos os endereços da rede de servidores, assim como o armazém.

```
root@tux14:/var/named# dig -b 192.168.2.1 loja1.qquma.pt +short
192.168.2.1
root@tux14:/var/named# dig -b 192.168.2.1 proxy.loja1.qquma.pt +short
192.168.2.2
root@tux14:/var/named# dig -b 192.168.2.1 qquma.pt +short
192.168.2.225
root@tux14:/var/named# dig -b 192.168.2.1 armazem.qquma.pt +short
192.168.2.193
root@tux14:/var/named# dig -b 172.16.1.14 armazem.qquma.pt +short
root@tux14:/var/named#
```

Figure 5.6: "Digs" na loja para endereços do sistema

Quando se faz dig para o endereço da loja, este devolve o próprio endereço da interface. Apesar de o endereço da loja não estar definido no ficheiro *database* do servidor DNS da rede de servidores, o pedido é reencaminhado para o slave que é neste caso o servidor DNS da loja. Este servidor tem na sua *database* os *DNS Records* dos domínios da loja, pelo que se confirma o correto funcionamento da tipologia *Master-Slave*.

O *Query* para o armazém é bem sucedido dado que o pedido é local. Externo não era possível resolver.

5.3 Armazém

No caso do armazém, este não consegue resolver nenhum pedido, pelo que todos são reencaminhados para a rede de servidores.

```
root@tux12:~# dig -b 192.168.2.193 armazem.qquma.pt +short
192.168.2.193
root@tux12:~# dig -b 192.168.2.193 proxy.loja1.qquma.pt +short
192.168.2.2
root@tux12:~# dig -b 192.168.2.193 qquma.pt +short
192.168.2.225
root@tux12:~# dig -b 192.168.2.193 google.com +short
142.250.184.174
```

Figure 5.7: "Digs" no armazém para endereços do sistema

Como de esperar, todas as *queries* são resolvidas corretamente como endereços locais, incluíndo a *query* para o próprio armazém.

6 Conclusão

Os objetivos inicialmente propostos foram atingidos. Foi elaborada um **projeto de endereçamento** para a rede que foi aprovado pelo docente.

Consequentemente, foram configuradas as diferentes **Vlans** e as respetivas *gateways* no router. Foram realizados testes `ping` que comprovaram a interconectividade entre os diferentes *hosts* na rede local.

A configuração do **NAT** foi realizada no intuito de apenas a DMZ conseguir aceder à internet. `Pings` a endereços públicos dos diferentes hosts apenas funcionaram na DMZ, como esperado.

Por fim, foram configurados os diferentes servidores DNS na rede. Através do comando `dig` foram feitas *queries* para diferentes domínios. Observamos que apenas computadores na rede local indicada (192.168.0.0/21) conseguiram resolver os endereços da rede de servidores, da loja, e do armazém.

A DMZ por sua vez, assim como outros domínios internos não autorizados (172.16.0.0/21), apenas conseguiram resolver os domínios da DMZ, não sendo capazes de aceder nem à loja nem ao armazém. Isto vai de encontro aos *DNS Records* que foram introduzidos para as diferentes zonas, pelo que mais uma vez se comprovou o funcionamento dos servidores DNS.

Em suma, este trabalho permitiu-nos adquirir novos conhecimentos na gestão e configuração de redes empresariais, aprofundando o nosso domínio na área do **Software Defined Network**.

Bibliografia

[1] *Creating Ethernet VLANs on Catalyst Switches*. Cisco Documentation. [Visitado 04-2021]. URL: \url{https://www.cisco.com/c/en/us/support/docs/lan-switching/vlan/10023-3.html}.

[2] *Configuring Routing Between VLANs with IEEE 802.1Q Encapsulation*. Cisco Documentation. [Visitado 04-2021]. URL: \url{https://www.cisco.com/en/US/docs/ios/lanswitch/configuration/guide/lsw_cfg_vlan_encap.html}.

[3] *Advanced DNS Features - Split DNS*. Bind 9 Documentation. [Visitado 04-2021]. URL: \url{https://bind9.readthedocs.io/en/v9_16_4/advanced.html#split-dns}.

[4] *BIND 9 Configuration Reference - Configuration File Grammar*. Bind 9 Documentation. [Visitado 04-2021]. URL: \url{https://bind9.readthedocs.io/en/latest/reference.html}.

[5] *Network Address Translation (NAT)*. Cisco Documentation. [Visitado 04-2021]. URL: \url{https://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/26704-nat-faq-00.html}.