Faculté des Sciences
Aix*Marseille Université

25/11/2016
Sylvain DE BARROS
Noé GOUDIAN

Projet Réseau

1. Configuration Réseau

On applique la configuration décrite à l'aide de fichiers ansible.

2. L'interface virtuelle TUN

2.2. Configuration de l'interface

Configurer l'interface tun0 avec l'adresse 172.16.2.1, mettre un masque adéquat. Ecrire un script configure-tun.sh reprenant vos commandes de configuration :

Il ne faut pas que notre nouveau réseau pour les tunnels entrent en collision avec les réseaux existants (LAN1, LAN2, LAN3, LAN4), on choisit donc de donner comme adresse au réseau du tunnel 172.16.2.0/28. Ce script est fourni dans l'archive sous le nom `configure-tun.sh`

Routage : Suite à la disparition tragique de VM2-6, faut-il modifier les informations de routage sur VM1 ? ou sur VM1-6 ?

Il faut modifier la route de VM1, sa nouvelle passerelle pour les réseaux LAN4 et LAN2 sera l'adresse de VM1-6 (qui accédera au tunnel) 172.16.2.156. Il faut également modifier le routage de VM1-6, sa passerelle pour les réseaux LAN2 et LAN4 sera le tunnel, 172.16.2.1.

Faire un `ping 172.16.2.1.` Donner la capture sur `tun0` (avec wireshark). Que constatez-vous ?

Il n' y a aucun paquet capturé. En effet les requêtes ping ne vont pas traverser l'interface tun0 car soit elle viennent de la même machine VM1-6 et alors elle ne vont pas 'descendre la pile' jusqu'aux interfaces; soit elles viennent de l'extérieur mais pas par le réseau sur lequel se trouve tun0 et vont donc entrer et sortir par une autre interface. Il est donc normal que la capture sur tun0 ne donne rien.

Faire ping6 172.16.2.10. Que constatez-vous ?

On voit passer des paquets (des requêtes mais pas de réponses). En effet, lorsque l'on adresse un ping à une machine qui appartient au sous réseau de tun0, c'est tun0 qui va

servir de passerelle pour tenter d'atteindre cette machine, et les requêtes vont donc sortir par tun0. En revanche, aucune machine n'ayant pour adresse 172.16.2.10, il n'y aura pas de réponse.

2.3. Récupération des paquets

Refaire ping6 172.16.2.1 puis ping6 172.16.2.10. Comparer et expliquer. Quel type de trafic voyez-vous? Refaire une capture avec wireshark dans le second cas et comparer avec ce qui est obtenu par votre programme test_iftun :

On réalise une capture sur le réseau avec notre programme, d'un ping sur 172.16.2.11:

```
m1reseaux@61vm1-6:/mnt/partage/C$ sudo ./iftun tun0 | hexdump -C
00000000  00 00 08 00 45 00 00 54  79 7e 40 00 40 01 64 fe  |....E..Ty~@.@.d.|
00000010  ac 10 02 01 ac 10 02 0b  08 00 f2 90 0b fd 00 01  |................|
00000020  89 3e 38 58 00 00 00 00  6f 07 0a 00 00 00 00 00  |.>8X....o.......|
00000030  10 11 12 13 14 15 16 17  18 19 1a 1b 1c 1d 1e 1f  |................|
00000040  20 21 22 23 24 25 26 27  28 29 2a 2b 2c 2d 2e 2f  | !"#$%&'()*+,-./|
00000050  30 31 32 33 34 35 36 37                           |01234567
```

On retrouve dans la capture les informations de niveau IP, le paquet ICMP, information de paquet.

Et simultanément sur Wireshark pour vérifier :

```
No.     Time           Source                Destination           Protocol Length Info
       1 0.000000000  172.16.2.1            172.16.2.11           ICMP    84      Echo (ping)
request  id=0x0bfd, seq=1/256, ttl=64 (no response found!)

Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
Raw packet data
Internet Protocol Version 4, Src: 172.16.2.1 (172.16.2.1), Dst: 172.16.2.11 (172.16.2.11)
Internet Control Message Protocol

0000  45 00 00 54 79 7e 40 00 40 01 64 fe ac 10 02 01   E..Ty~@.@.d.....
0010  ac 10 02 0b 08 00 f2 90 0b fd 00 01 89 3e 38 58   .............>8X
0020  00 00 00 00 6f 07 0a 00 00 00 00 00 10 11 12 13   ....o...........
0030  14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23   ............ !"#
0040  24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33   $%&'()*+,-./0123
0050  34 35 36 37                                       4567
```

On observe les mêmes résultats, à la différence près que la capture avec notre programme ajoute 2 champs de deux octets chacuns, "flags" et "proto", au début.

A quoi sert l'option IFF_NO_PI ? Que ce passe-t-il si vous ajoutez cette option lors de la création de l'interface ?

Le drapeau IFF_NO_PI ("Do not provide packet information") permet de ne pas avoir les champs flags et proto en début de paquet.

```
00000000  45 00 00 54 e1 6a 40 00  40 01 fd 11 ac 10 02 01  |E..T.j@.@.......|
00000010  ac 10 02 0b 08 00 52 10  0d 22 00 01 82 42 38 58  |......R.."...B8X|
00000020  00 00 00 00 1a 5f 05 00  00 00 00 00 10 11 12 13  |....._..........|
00000030  14 15 16 17 18 19 1a 1b  1c 1d 1e 1f 20 21 22 23  |............ !"#|
00000040  24 25 26 27 28 29 2a 2b  2c 2d 2e 2f 30 31 32 33  |$%&'()*+,-./0123|
00000050  34 35 36 37                                       |4567|
```

Et dans Wireshark :

```
No.     Time           Source                Destination           Protocol Length Info
       1 0.000000000  172.16.2.1            172.16.2.11           ICMP    84      Echo (ping)
request  id=0x0d22, seq=1/256, ttl=64 (no response found!)

Frame 1: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
Raw packet data
Internet Protocol Version 4, Src: 172.16.2.1 (172.16.2.1), Dst: 172.16.2.11 (172.16.2.11)
Internet Control Message Protocol

0000  45 00 00 54 e1 6a 40 00 40 01 fd 11 ac 10 02 01   E..T.j@.@.......
0010  ac 10 02 0b 08 00 52 10 0d 22 00 01 82 42 38 58   ......R.."...B8X
0020  00 00 00 00 1a 5f 05 00 00 00 00 00 10 11 12 13   ....._..........
0030  14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23   ............ !"#
0040  24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33   $%&'()*+,-./0123
0050  34 35 36 37                                         4567
```

Cette fois on a bien la même capture, en direct et avec Wireshark.