Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Réseaux Avancés II

TD n°3

1 – On veut programmer un outil d'« *OS fingerprint* », c-à-d qui découvre la nature de l'OS utilisé par la machine, en se basant sur l'observation suivante :

OS	TTL	taille fenêtre TCP
Linux kernel 2.x	64	5840
Android/Chrome OS	64	5720
Windows XP	128	65536
Windows 7	128	4128
Cisco	255	4128
FreeBSD	64	65535

On remarque que lors de l'établissement de la communication TCP, chaque machine envoie la valeur de sa fenêtre, « window », et la valeur du TTL.

Suivant les différentes combinaisons de valeurs, on peut identifier l'OS utilisé par la machine qui a envoyé le paquet.

a. Si on peut observer les paquets échangés lors de l'établissement de la communication TCP, est-ce que l'on peut identifier l'OS :
   - ◇ les deux simultanément ? *Oui, lors du «* handshake *» TCP, le client transmet un paquet SYN avec un TTL et sa taille de fenêtre, auquel répond le serveur avec un paquet SYM/ACK avec un TTL et sa propre taille de fenêtre*
     Le TTL peut être ajusté à la valeur la plus proche en considérant qu'il est rare de passer par plus de 10 routeurs sur Internet.

b. Est-ce qu'il est possible d'identifier l'OS d'une machine de manière active, c-à-d en provoquant l'échange de paquets TCP ? *Oui, il suffit d'établir une connexion TCP avec la machine cible pour obtenir un paquet contenant TTL et taille de fenêtre de la machine cible*.

c. Écrivez un programme Python utilisant Scapy qui réalise le travail d'identification des OS en écoutant le trafic circulant dans un réseau (on considérera que l'on est capable d'intercepter tout les paquets échangés).

```
#!/usr/bin/python3

from scapy.all import *

dico_os = { 'Cisco' : (255, 4128), 'Linux' : (64,58140) }

def traiter_trame(t):
    if TCP not in t:
        return
    ttl = t[IP].ttl
    window = t[TCP].window
    trouve = False
    for os in dico_os.keys():
        os_ttl,os_win = dico_os[os]
        if ((os_ttl == ttl) and (os_win == window)):
            print ("La machine %s est un %s"%(t[IP].src,os))
            trouve = True
    if (trouve == False):
         print("OS Inconnu")

#sniff(count = 0, prn = traiter_trame, filter = "tcp")
# Pour tester directement le programme :
traiter_trame(Ether()/IP(src='127.128.129.130',ttl=255)/TCP(window = 4128))
traiter_trame(Ether()/IP(src='127.128.129.130',ttl=64)/TCP(window = 4128))
traiter_trame(Ether()/IP(src='127.128.129.130',ttl=64)/TCP(window = 58140))
```

2 – Analysez la trame suivante :

```
0000   78 92 9C E4 5B 83 DE 30  05 DD 1C B4 86 DD 60 00   x...[..0......`.
0010   00 00 00 18 2C 2F 26 07  53 00 00 60 00 5C DC 30   ....,/&.S..`.\.0
0020   05 FF FE DD 1C B4 20 01  41 D0 FE 0B 85 00 A8 E2   ...... .A.......
0030   43 FF FE 2A AC 3D 11 00  00 11 D1 56 C0 4E 00 00   C..*.=.....V.N..
0040   00 00 03 77 77 77 06 67  6F 6F 67 6C 65 03         ...www.google.
```

a. Que contient la trame ? *L'analyse sous Scapy du paquet donne :*

```
<Ether  dst=78:92:9c:e4:5b:83 src=de:30:05:dd:1c:b4 type=IPv6
|<IPv6  plen=None nh=Fragment Header hlim=47 src=2607:5300:60:5c:dc30:5ff:fedd:1cb4
dst=2001:41d0:fe0b:8500:a8e2:43ff:fe2a:ac3d
|<IPv6ExtHdrFragment  nh=UDP offset=2 m=1 id=3512123470
|<Raw  load='\x00\x00\x00\x00\x03www\x06google\x03' |>>>>
```

b. Est-elle passée par un routeur ?
Oui, le « Hop Limit *», HL, est inférieur à 64.*

c. Est-ce que les adresses IPv6 ont été obtenues par « **auto-configuration** » ?
- ◇ *pour **l'adresse source** :*
  - ⋆ *L'adresse source est* `2607:5300:60:5c:dc30:5ff:fedd:1cb4` *ce qui donne l'identifiant machine sur 64bits* `dc30:5ff:fedd:1cb4`.
  - ⋆ *L'adresse MAC source est* `de:30:05:dd:1c:b4`.
    On remarque la présence du motif `fffe` *au milieu de l'adresse MAC et l'inversion du 7ème bit :*

    Adresse MAC — *Premier octet :* `de`

						d	e
1	1	0	1	1	1	1	0
8	4	2	1	8	4	2	1
1	2	3	4	5	6	7	8

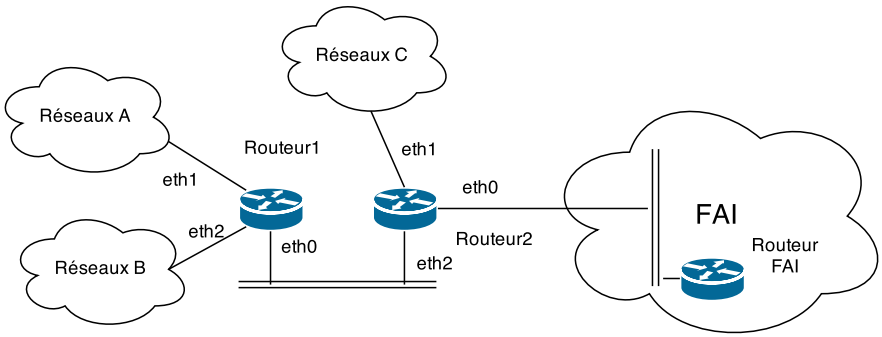
    Identifiant machine — *Premier octet :* `dc`

						d	c
1	1	0	1	1	1	0	0
8	4	2	1	8	4	2	1
1	2	3	4	5	6	7	8

    ⟹*adresse obtenue par **auto-configuration** à partir de l'adresse MAC locale de la machine associée.*
- ◇ *pour **l'adresse destination** :*
  - ⋆ *l'adresse destination est* `2001:41d0:fe0b:8500:a8e2:43ff:fe2a:ac3d` *elle ne fait pas partie du même réseau que l'adresse source ;*
  - ⋆ *l'adresse source a été obtenue par auto-configuration par rapport à l'adresse MAC présente dans la trame ⟹ le réseau local est de préfixe globale* `2607:5300:60:5c::/64` *;*
  - ⋆ *l'adresse destination est extérieure au réseau locale ⟹ l'adresse MAC de destination* `78:92:9c:e4:5b:83` *doit être celle du routeur de sortie :*

d. Peut-on apprendre des informations sur la nature du matériel en réception et/ou en envoi ?

3 – Soit le réseau d'entreprise suivant, découpé en 3 sous-réseaux :
- ▷ Réseaux A « `172.16.200.0/24` » ;
- ▷ Réseaux B « `172.16.100.0/24` » ;
- ▷ Réseaux C « `172.16.300.0/24` ».



et la configuration suivante :

Routeur 1
- ☐ `eth0:192.168.1.10/24` ;
- ☐ `eth1:172.16.200.0/24` ;
- ☐ `eth2:172.16.100.0/24` ;

Routeur 2
- ☐ `eth0:193.50.123.32/24` ;
- ☐ `eth1:172.16.300.0/24` ;
- ☐ `eth2:192.168.1.20/24` ;

Le routeur FAI possède l'adresse IP « `193.50.123.254` ».

Pour votre **migration IPv6**, vous obtenez le préfixe `2003:8b:0c::/56` de la part de votre FAI.
Le routeur du FAI possède l'adresse IPv6 `2003:8b:0c:ff:ff:ff:ff:ff/64`

a. En IPv6, comment le routeur « *Routeur1* » peut découvrir **automatiquement** « *Routeur2* » ?
En utilisant l'adresse de multicast `ff02::2` *qui désigne l'ensemble des routeurs connectés au réseau local (un paquet envoyé vers cette adresse multicast est traité par l'ensemble des routeurs qui le reçoivent).*

b. Est-ce qu'il est nécessaire que le **réseau d'interconnexion** des routeurs disposent d'un préfixe global ?
Non, ils peuvent utiliser des adresses « lien local *» de préfixe* `fe80`.
Donnez l'adresse IPv6 de type « *SLAAC* » pour l'interface `eth0` de « *Routeur1* » d'adresse MAC `38:60:77:4f:56:eb`.
On peut utiliser Linux pour faire le SLAAC en « link-local *» :*

```
xterm
> sudo ip l add dev toto type dummy
> sudo ip l set dev toto address 38:60:77:4f:56:eb
> sudo ip l set dev toto up
> ip a
12: toto: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/ether 38:60:77:4f:56:eb brd ff:ff:ff:ff:ff:ff
    inet6 fe80::3a60:77ff:fe4f:56eb/64 scope link
       valid_lft forever preferred_lft forever
```

D'où l'adresse globale finale : `2003:8b:0c::3a60:77ff:fe4f:56eb`.

c. Donnez le **plan d'adressage IPv6** pour les réseaux A, B et C.
On dispose d'un préfixe réseau global en `\56:`

							/56 /64
20	*03*	*8b*	*0c*	*00*	*00*	*00*	*00*

On a donc 1 octet, soient deux chiffres hexadécimaux que l'on peut choisir.
Ce qui donne :

Réseau	*adresse*
ResA	`2003:8b:0c:0001::/64`
ResB	`2003:8b:0c:0002::/64`
ResC	`2003:8b:0c:0003::/64`

d. Donnez la **table de routage IPv6** de « *Routeur2* ».

destination	Next Hop	
default	2003:8b:0c:ff:ff:ff:ff:ff	routeur FAI
ResA	fe80:3a60:77ff:fe4f:56eb	routeur R1
ResB	fe80:3a60:77ff:fe4f:56eb	routeur R1
ResC	2003:008b:000c:0003:ffff:ffff:ffff	