TP0 Guillaume Sanchez

Partie 1 : Analyse réseaux avec Wireshark

NaN

Partie 2 : Injection de trafic avec l’outil Scapy

Exercice 3 :

```
>>> a = IP(dst="192.168.1.1", src="10.10.10.1")/ICMP()
```

```
>>> ls(a)
version    : BitField  (4 bits)                  = 4               ('4')
ihl        : BitField  (4 bits)                  = None            ('None')
tos        : XByteField                          = 0               ('0')
len        : ShortField                          = None            ('None')
id         : ShortField                          = 1               ('1')
flags      : FlagsField                          = <Flag 0 ()>     ('<Flag 0 ()>')
frag       : BitField  (13 bits)                 = 0               ('0')
ttl        : ByteField                           = 64              ('64')
proto      : ByteEnumField                       = 1               ('0')
chksum     : XShortField                         = None            ('None')
```

```
src        : SourceIPField                         = '10.10.10.1'    ('None')
dst        : DestIPField                           = '192.168.1.1'   ('None')
options    : PacketListField                       = []              ('[]')
--
type       : ByteEnumField                         = 8               ('8')
code       : MultiEnumField (Depends on 8)         = 0               ('0')
chksum     : XShortField                           = None            ('None')
id         : XShortField (Cond)                    = 0               ('0')
seq        : XShortField (Cond)                    = 0               ('0')
ts_ori     : ICMPTimeStampField (Cond)             = None            ('38407079')
ts_rx      : ICMPTimeStampField (Cond)             = None            ('38407079')
ts_tx      : ICMPTimeStampField (Cond)             = None            ('38407079')
gw         : IPField (Cond)                        = None            ("'0.0.0.0'")
ptr        : ByteField (Cond)                      = None            ('0')
reserved   : ByteField (Cond)                      = None            ('0')
length     : ByteField (Cond)                      = None            ('0')
addr_mask  : IPField (Cond)                        = None            ("'0.0.0.0'")
nexthopmtu : ShortField (Cond)                     = None            ('0')
unused     : MultipleTypeField (ShortField, IntField, StrFixedLenField) = b''               ("b''")
extpad     : _ICMPExtensionPadField (Cond)         = None            ("b''")
ext        : _ICMPExtensionField (Cond)            = None            ('None')
```

```
>>> a = IP(dst="192.168.1.1")/ICMP(type=8, code=0)
>>> a.summary()
'IP / ICMP 10.25.32.159 > 192.168.1.1 echo-request 0'
>>> res =sr(a)
Begin emission
.
Finished sending 1 packets
.*
```

```
Received 3 packets, got 1 answers, remaining 0 packets
```

```
>>> res[0].summary()
IP / ICMP 10.25.32.159 > 192.168.1.1 echo-request 0 ==> IP / ICMP 192.168.1.1 > 10.25.32.159 echo-reply 0 / Paddi
```

```
>>> sendp(Ether(dst="08 :11 :96 :f6 :42 :12")/IP(dst="192.168.1.1") / ICMP())
.
Sent 1 packets.
>>> a=Ether()/IP()/ICMP()
>>> sendp(a)
.
Sent 1 packets.
```

```
>>> sendp(IP(dst="72.14.207.99")/TCP(dport=80,flags="S"))
.
Sent 1 packets.
>>> sendp(IP(dst="192.168.1.1")/TCP(sport=666,dport=(440,443),flags="S"))
....
Sent 4 packets.
```

```
>>> hexdump(a)
0000  FF FF FF FF FF FF 00 00 00 00 00 00 08 00 45 00  ..............E.
0010  00 1C 00 01 00 00 40 01 7C DE 7F 00 00 01 7F 00  ......@.|.......
0020  00 01 08 00 F7 FF 00 00 00 00                    ..........
```

Exercice 4 :

```
>>> send( fragment(IP(dst="10.0.0.5")/ICMP()/("X"*1472)) )
.
Sent 1 packets.
>>> send( fragment(IP(dst="10.0.0.5")/ICMP()/("X"*1473)) )
..
Sent 2 packets.
```

La première commande, la machine reçoit 1 seul paquet.
La deuxième commandes, la machine reçoit 2 paquets en fragments?.

L'explication technique repose sur la taille maximale d'un paquet sur un réseau Ethernet, appelée MTU (Maximum Transmission Unit), qui est généralement de 1500 octets.

- Pour le premier paquet (1472 octets de données) :
  - Données ("X" * 1472) + Entête ICMP (8 octets) + Entête IP (20 octets) = 1500 octets.
  - Le total est exactement égal à la MTU. Le paquet n'a pas besoin d'être découpé, il est envoyé tel quel.
- Pour le deuxième paquet (1473 octets de données) :
  - Données ("X" * 1473) + Entête ICMP (8 octets) + Entête IP (20 octets) = 1501 octets.
  - Le total dépasse la MTU de 1500 octets. Scapy, via la fonction fragment(), est donc obligé de diviser ce message en deux fragments IP pour qu'ils puissent circuler sur le réseau.

```
>>> pkts = sniff(count=10)
>>> pkts.show()
0000 Ether / IPv6 / UDP / mDNS Ans [b'']
0001 Ether / IP / UDP / mDNS Ans [b'']
```

```
0002 Ether / IP / UDP / DNS Qry b'www.googleapis.com.'
0003 Ether / IP / UDP / DNS Qry b'www.googleapis.com.'
0004 Ether / IP / UDP 10.25.32.159:41667 > 172.217.22.138:https / Raw
0005 Ether / IP / UDP 10.25.32.159:41667 > 172.217.22.138:https / Raw
0006 Ether / IP / UDP 10.25.32.159:41667 > 172.217.22.138:https / Raw
0007 Ether / IP / UDP 172.217.22.138:https > 10.25.32.159:41667 / Raw
0008 Ether / IP / UDP 172.217.22.138:https > 10.25.32.159:41667 / Raw
0009 Ether / IP / UDP 172.217.22.138:https > 10.25.32.159:41667 / Raw
```

```
>>> pkts = sniff(filter="icmp and host 192.168.1.1", count = 2)
>>> pkts.show()
0000 Ether / IP / ICMP 192.168.1.10 > 192.168.1.1 echo-request 0 / Raw
0001 Ether / IP / ICMP 192.168.1.1 > 192.168.1.10 echo-reply 0 / Raw
```

Exercice 5 :

```
>>> sendp(IP(dst="10.1.1.5", ihl=2, version=3)/ICMP())
.
Sent 1 packets.
```

```
>>> sendp(IP(src="192.168.1.1",dst="10.1.1.5")/ ICMP()/("X"*1472),loop=1)
.................................................................................................
.................................................................................................
.................................................................................................
.................................................................................................
.................................................................................................
.................................................................................................
..............................................................................................^C
```

```
Sent 676 packets.
```

```
>>> res,unans = sr( IP(dst="192.168.1.1")/TCP(flags="S", dport=(1,1024)))
Begin emission
.**.*............................
Finished sending 1024 packets
.................................................................................................................
.......INFO: DNS RR prematured end (ofs=5, len=5)
.....................................................INFO: DNS RR prematured end (ofs=5, len=5)
...................................................................INFO: DNS RR prematured end (ofs=5, len=5)
.................................................................................................................
Received 3291 packets, got 3 answers, remaining 1021 packets
>>> unans[1].summary()
'IP / TCP 10.25.32.159:ftp_data > 192.168.1.1:2 S'
```

Exercice 6 :

```
>>> res,unans=traceroute(["www.voila.com"],maxttl=20)
Begin emission
********
Finished sending 20 packets
***********
Received 19 packets, got 19 answers, remaining 1 packets
   69.49.101.52:tcp80
1  10.25.32.1      11
2  192.168.1.254   11
3  212.30.97.108   11
4  77.192.212.93   11
```

```
5  194.6.145.206   11
6  62.115.154.22   11
7  62.115.118.62   11
8  62.115.133.238  11
10 62.115.141.37   11
11 62.115.141.35   11
12 62.115.45.29    11
13 69.165.84.9     11
14 38.132.50.134   11
15 216.55.190.77   11
16 69.49.100.146   11
17 69.49.101.52    SA
18 69.49.101.52    SA
19 69.49.101.52    SA
20 69.49.101.52    SA
```

```
>>> res.show()
   69.49.101.52:tcp80
1  10.25.32.1      11
2  192.168.1.254   11
3  212.30.97.108   11
4  77.192.212.93   11
5  194.6.145.206   11
6  62.115.154.22   11
7  62.115.118.62   11
8  62.115.133.238  11
10 62.115.141.37   11
11 62.115.141.35   11
12 62.115.45.29    11
13 69.165.84.9     11
```

```
14 38.132.50.134   11
15 216.55.190.77   11
16 69.49.100.146   11
17 69.49.101.52    SA
18 69.49.101.52    SA
19 69.49.101.52    SA
20 69.49.101.52    SA
```

Les commandes de types graph ne fonctionne pas sur ma machine.

```
>>> rep,non_rep=sr( IP(dst='209.85.143.100', ttl=(1,25)) / ICMP(), timeout=1 )
...: rep.summary()
Begin emission
..********
Finished sending 25 packets
*.************...
Received 27 packets, got 21 answers, remaining 4 packets
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 10.25.32.1 > 10.25.32.159 time-exceeded ttl-
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 192.168.1.254 > 10.25.32.159 time-exceeded t
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 194.6.146.51 > 10.25.32.159 time-exceeded tt
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 81.65.205.33 > 10.25.32.159 time-exceeded tt
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 212.30.97.108 > 10.25.32.159 time-exceeded t
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 81.65.44.57 > 10.25.32.159 time-exceeded ttl
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 72.14.233.195 > 10.25.32.159 time-exceeded t
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 108.170.255.238 > 10.25.32.159 time-exceeded
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 192.178.81.229 > 10.25.32.159 time-exceeded
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 142.250.238.94 > 10.25.32.159 time-exceeded
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.142.24 > 10.25.32.159 time-exceeded t
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
IP / ICMP 10.25.32.159 > 209.85.143.100 echo-request 0 ==> IP / ICMP 209.85.143.100 > 10.25.32.159 echo-reply 0 /
```