

# Master 1<sup>ère</sup> année

## Réseaux Avancés II

Corrections TP n°1

Introduction à Scapy

### **■ ■ ■** Contrôle d'erreur

- 1 Différence « checksum » et « CRC » :
  - a. Créez à l'aide des fonctions et objets disponibles dans Scapy deux trames Ethernet contenant :
    - 1. un paquet UDP contenant les données : '\x00\x00';
    - 2. un paquet UDP contenant les données : '\xFF\xFF';
  - b. Comparez les valeurs *calculées* des checksums présents dans la couche UDP: sont ils identiques? Pourquoi?
    - Les valeurs sont identiques, car avec le checksum, l'addition de 65535 est équivalent à ajouter 0!
  - c. Calculez le CRC-32 de chacune de ces trames : sont-ils différents ? Pourquoi ? Les valeurs sont différentes car le CRC est capable de détecter une modification de la trame d'une longueur de 16 bits.

#### ■■■ Détection de trafic d'audit

2 – Écrire un programme utilisant Scapy permettant de détecter si on est la cible d'un scannage de port :

```
1 #!/usr/bin/python
 2
3 from scapy.all import *
  traces = {}
 7 # traces est un dico de dico {@IPsrc:{port:temps, ...port:temps}, ...}
 8 def verifier_attaque(a,t):
 9
       if len(t)<10:
           return
11
       temps_min = min(t.values())
       temps_max = max(t.values())
12
       if (temps_max-temps_min) < 10:</pre>
13
14
           print ("Attaque possible depuis %s"%a)
15
16 def traiter_trame(t):
17 if (not TCP in t)
       if (not TCP in t):
18
19
           return
       if (t.sprintf("%TCP.flags%")!='S'):
20
21
           return
       adresse_source = t[IP].src
22
       port_destination = t[TCP].dport
23
       temps = t.time
       if (adresse_source in traces):
25
           table_ports = traces[adresse_source]
26
           table_ports[port_destination] = temps
27
           print (traces)
           verifier_attaque(adresse_source,table_ports)
28
29
       else :
           traces[adresse_source] = {port_destination:temps}
30
32 sniff(count=0, filter="tcp", prn=traiter_trame)
```

### 3 - Réalisation d'un serveur de « Port Knocking ».

Vous écrirez le programme Python utilisant Scapy et réalisant :

- \* la détection de cette séquence de tentative de connexion correspondant à du « port knocking » ;
- \* l'autorisation à l'aide de NetFilter de la connexion vers l'application (le travail du «PK daemon »):
  - la séquence de tentative de connexion TCP est la suivante: [2027, 3230, 2001, 17377];
  - l'application attend en TCP sur le port 16400;
  - le serveur hébergeant l'application et votre programme possède l'@IP 192.168.1.137.

```
1 #!/usr/bin/python
 3 from scapy.all import *
 4 import commands
6 sequence = [2027, 3230, 2001, 17377]
7 adresse_serveur = '192.168.1.137'
8 adresse_client = ''
9 port_application = 16400
10 rang_sequence = 0
11
12 def traiter_paquet(p):
    global rang_sequence
    if (p.sprintf('%TCP.flags%') == 'S'):
14
15
        if ((rang_sequence == 0) and (p[TCP].dport == sequence[0])):
         adresse_client = p[IP].src
16
17
      if ((p[TCP].dport == sequence[rang_sequence]) and (p[IP].src == adresse_client)):
18
        rang_sequence += 1
         print ("Knock...", rang_sequence)
19
20
         if (rang_sequence == len(sequence)):
21
            rang_sequence = 0
commands.getoutput('sudo iptables -A INPUT -s %s -p tcp --dport %d -m state --state NEW, ESTABLISHED -j ACCEPT'% (adresse_client,port_application))
23 sniff(filter="(dst host %s) and tcp"%adresse_serveur, prn=traiter_paquet)
```