Groupe 4 — Attaques sur TCP avec scapy

L'objectif est d'expérimenter des attaques réseau basées sur TCP à l'aide du paquet python *scapy*.
Construire le réseau est composé de 3 PC connectés à un hub (Les membre de votre groupe).
L'hôte scapy est celui depuis lequel vous expérimenterez les attaques visant à perturber les connexions telnet entre le client et le serveur. Le paquet scapy y est déjà installé.
Vous trouverez en annexe de ce projet des rappels sur scapy. Votre compte-rendu devra contenir les codes des scripts ainsi que des copies d'écran et explications des tests effectués.

Question 1 — Travaux préliminaires

I 1.1 Ouvrez le projet et démarrez tous les équipements.
I 1.2 Ouvrez une session root sur chaque hôte.
I 1.3 Attribuez les IP suivantes aux hôtes :
- client : 10.0.0.1/24
- serveur : 10.0.0.2/24
- scapy : 10.0.0.100/24

Dans la suite, on utilisera telnet pour tester nos attaques. Voici, pour rappel, les instructions pour l'utiliser.

I 1.4 Sur le serveur : lancez le service inetd :

```
$ /etc/init.d/inetutils-inetd restart
```

I 1.5 Sur le client : ouvrez une session telnet sur le serveur :

```
$ telnet 10.0.0.2
```

Connectez vous avec le compte `student` (mot de passe = `student`). Ceci va ouvrir un shell sur le serveur. Vous pouvez ensuite le fermer avec Ctrl+D ou `exit`.

Question 2 — L'attaque SYN flood

Le script flood.py effectue une attaque de type SYN flood sur une destination quelconque. Le script prend deux arguments : l'adresse IP et le numéro de port à attaquer. Le script envoie ensuite des paquets en continu, sans s'arrêter.

I 2.1 Écrivez le script `flood.py`.

Indications.

— En théorie, l'attaquant utilise une IP et un port source choisis aléatoirement comme nous l'avons vu en TD. Pour éviter d'envoyer des paquets vers l'extérieur, vous utiliserez une IP source aléatoire dans le réseau IP de vos 3 hôtes.

— Utilisez la fonction randint(min, max) du paquet random pour générer un nombre aléatoire dans [*min*,*max*]. Pour tester votre script :

I 2.2 Sur le client : si une session telnet est ouverte, fermez la.
I 2.3 Sur scapy : lancez l'attaque sur le serveur. Le port utilisé par le service telnet est le port 23.
I 2.4 Pour vérifier que l'attaque fonctionne :
- Sur le serveur : lancez la commande suivante qui affiche l'état des connexion TCP :

```
$ netstat -tna
```

Vous devriez voir de nombreuses connexions dans l'état SYN_RECV. Ce sont des connexions non finalisées pour lesquelles seul le paquet SYN a été reçu.

— Sur le client : ouvrez une session telnet sur le serveur. Soit la commande telnet échoue avec un message indiquant Connection timed out. Soit elle se fait mais difficilement : le client reste longtemps dans l'état Trying ... avant d'afficher le message Connected to ....

Question 3 — L'attaque TCP reset

Le script reset.py effectue une attaque de type TCP reset. Le script prend en argument l'adresse IP à attaquer et un numéro de port. Le script capture les paquets TCP en provenance de l'IP passée en argument et destinés au port passé en argument. À chaque fois qu'un tel paquet est reçu, le script envoie un paquet TCP pour mettre fin à la connexion.

I 3.1 Écrivez le script `reset.py`.

Pour tester votre script :

I 3.2 Sur le client : ouvrez une session telnet sur le serveur.
I 3.3 Sur scapy : lancez l'attaque sur le client.
I 3.4 Sur le client : vérifez que la session est bien fermée si on tape sur une touche.

Question 4 — Le vol de session TCP

Le script hijack.py effectue une attaque de vol de session TCP. Le script prend en argument l'adresse IP du serveur à attaquer. Il attend ensuite la capture d'un paquet de données telnet en destination (ou en provenance) de ce serveur avant d'envoyer son paquet d'attaque. Dans en premier temps, on utilisera l'attaque pour que l'attaquant crée le fichier /home/student/hacked.txt contenant la phrase *"Sabotage!!!"*.

I 4.1 Écrivez le script `hijack.py`.

Indications. Il faut utiliser la classe Raw de scapy pour encapsuler des données quelconques dans un PDU. Par exemple, le code ci-dessous crée un paquet TCP contenant les données telnet ls /tmp :

```
p = IP() / TCP(dport=23) / Raw("ls\n/tmp")
```

Pour tester votre script :

I 4.2 Sur le client : ouvrez un nouveau terminal :

```
$ xterm &
```

Dans ce nouveau terminal, ouvrez une session telnet sur le serveur.

I 4.3 Sur scapy : lancez l'attaque sur le serveur.
I 4.4 Sur le client : tapez une touche dans le terminal telnet. Ceci va déclencher l'envoi de paquets de données entre le client et le serveur et donc l'attaque. Si celle-ci s'est bien déroulée, la session telnet ne devrait plus répondre. Quel est le problème?
I 4.5 Sur le serveur : vérifiez que le fichier /home/student/hacked.txt a bien été créé.

En utilisant l'attaque de vol de session, l'attaquant peut aussi forcer le serveur à entrer en contact avec lui, par exemple, pour lui faire envoyer le contenu d'un fichier. Nous allons pour cela utiliser la commande nc qui permet,

entre autres, de démarrer un serveur TCP. Nous allons dans un premier temps nous familiariser avec cette commande :

I 4.6 Sur scapy : lancez un serveur en écoute (option **-l** pour listen) sur le port 12345 :

```
$ nc -l -p 12345
```

I 4.7 Sur le serveur : ouvrez une connexion TCP sur le port 12345 de scapy pour y envoyer des données quelconques (le mot `hello` par exemple) :

```
$ echo hello | telnet 10.0.0.100 12345
```

(Le port par défaut utilisé par la commande `telnet` est 23 mais on peut l'utiliser pour contacter n'importe quel port.)

I 4.8 Sur scapy : vérifiez que le mot `hello` a bien été écrit dans le terminal. Le serveur n'est alors plus en écoute.

Passons maintenant à la modification du script.

I 4.9 Modifiez le script hijack.py (il suffit juste de modifier la commande que l'attaquant fait exécuter au serveur) pour que le serveur envoie le contenu de son fichier /etc/passwd à l'attaquant sur le port 12345.

I 4.10 Testez votre modification en démarrant au préalable, dans un nouveau terminal de scapy, un serveur TCP en écoute sur le port 12345. Si l'attaque fonctionne le contenu du fichier /etc/passwd du serveur devrait s'afficher dans le terminal de scapy dans lequel s'exécute nc.

I 4.11 Enfin, documentez vous sur le principe du shell inversé (reverse shell) et modifiez le script hijack.py afin qu'il fasse exécuter au serveur l'ouverture d'un shell sur le port 12345 de scapy.

Question 5 — Le rapport de vol de session TCP sur les 3 machines

Annexe: Rappels sur scapy

```
# importer tout ce qu'il faut en début de fichier
from scapy.all import IP, ICMP, TCP, UDP, ...

# forger un PDU IP avec les valeurs par défaut
p = IP()

# méthode show pour afficher le contenu du PDU
p.show()

# forger des PDU avec des valeurs spécifiques
p = IP(src="10.0.2.1", dst="10.0.2.2", ttl=22)
p = TCP(sport=54, dport=789, flags="RSAFUP") # R=RST, S=SYN, A=ACK, F=FIN, U=URG, P=PSH

# forger des PDU encapsulant d'autres PDU avec l'opérateur /
p = IP() / TCP()
p = IP() / UDP() / DNS()
p = Ether() / IP() / TCP()

# envoi d'un PDU de niveau 3
p = IP() /
TCP() send(p)

# envoi d'un PDU de niveau 2
p = Ether()
/ IP() / TCP() sendp(p)

# envoi d'un PDU de niveau 3 et réception de la réponse
p = IP() / TCP()
reponse = sr1(p, timeout=2) # on attend 2 sec. max
if reponse is None: print("aucune réponse reçue")
else :
    reponse.show()

# envoi d'un PDU de niveau 2 et réception de la réponse
p = Ether() / IP() /
TCP()
reponse = srp1(p, timeout=2)
if reponse is None:
    ...

# tester qu'un PDU est bien présent dans un paquet
if ICMP in reponse:
    print("la réponse contient un PDU ICMP")

# accéder à un PDU if TCP in reponse:
pdu_tcp = reponse[TCP]
    print(pdu_tcp.flags)

# capturer des paquets
paquets = sniff(timeout=2)
print("liste des paquets capturés pendant 2 secondes :")
for paquet in paquets:
    paquet.show()

# capturer des paquets et appeler une fonction à chaque fois qu'un paquet est reçu
def affiche_paquet_recu(paquet):
    paquet.show()
sniff(timeout=10, prn=affiche_paquet_recu)
```