Travaux dirigés

Exercice 1 — Échange de données TCP

On considère dans cet exercice un client HTTP (10.0.0.10:53201) envoyant la requête HTTP :

```
GET  /index.html  HTTP/1.1
```

vers le serveur HTTP 10.0.0.100:80. On fait également les hypothèses suivantes :
— La requête HTTP a un en-tête de 200 octets et un corps vide.
— La réponse HTTP a un en-tête de 100 octets et le fichier demandé a une taille de 1400 octets.
— Le client et le serveur sont sur le même réseau physique Ethernet.
On rappelle aussi :
— en-tête IP = en-tête TCP = 20 octets
— HTTP est en mode non connecté.
Tracer sur un chronogramme les échanges de paquets observés entre le client et le serveur dans les scénarios suivants :

Q 1.1 Le serveur refuse la connexion du client (p.ex., si trop de requêtes sont actuellement en traitement).

Correction

```
[01]  C -> S : Flags=S, SEQ=x
[02]  S -> C : Flags=RA, ACK=x+1
```

◆

Q 1.2 Tout se déroule “normalement” (le serveur envoie le fichier au client et aucun paquet n’est perdu).

Correction

```
[01]  C -> S : Flags=S, SEQ=x
[02]  S -> C : Flags=SA, SEQ=y, ACK=x+1
[03]  C -> S : Flags=A, SEQ=x+1, ACK=y+1
[04]  C -> S : Flags=A, SEQ=x+1, ACK=y+1, Len=200        # requête GET
[05]  S -> C : Flags=A, SEQ=y+1, ACK=x+201
[06]  S -> C : Flags=A, SEQ=y+1, ACK=x+201, Len=1460     # en-tête de la réponse + 1360 o. HTML
[07]  C -> S : Flags=A, SEQ=x+201, ACK=y+1461
[08]  S -> C : Flags=A, SEQ=y+1461, ACK=x+201, Len=40    # 40 o. HTML
[09]  C -> S : Flags=A, SEQ=x+201, ACK=y+1501
[10]  C -> S : Flags=FA, ACK=y+1501
[11]  S -> C : Flags=FA, ACK=x+201
[12]  C -> S : Flags=A, ACK=y+1501
```

◆

Faites apparaître, pour chaque paquet envoyé, les différents champs de l’en-tête TCP utilisés dans le contrôle de l’échange.

Exercice 2 — Attaque TCP reset

Q 2.1 Rappeler le principe de l’attaque TCP reset.

Correction

Mettre fin à une connexion TCP en envoyant un paquet RST suite à la capture d’un paquet TCP. ◆

Q 2.2 Peut-on utiliser cette attaque pour perturber une connexion établie entre un client et un serveur SSH communiquant sur IP ? (On rappelle que les données SSH sont chiffrées.) Justifier.

Correction

Oui, car cette attaque se base uniquement sur les drapeaux et champs de l’en-tête TCP et ceux-ci ne sont pas chiffrés. Seul le contenu des données TCP l’est. ◆

Q 2.3 Supposons qu’un attaquant arrive à capturer le paquet suivant :

```
1.2.3.4:12345 -> 2.3.4.5:23456 : Flags=A, SEQ=45243, ACK=85763, Len=200
```

Où `Len` est la longueur des données TCP.

Quel(s) paquet(s) peut-il envoyer pour perturber la connexion ? Donnez les contenus des champs significatifs.

Correction

`1.2.3.4:12345 -> 2.3.4.5:23456 : Flags=R, SEQ=45443`

ou

`2.3.4.5:23456 -> 1.2.3.4:12345 : Flags=R, SEQ=85763`

◆

Exercice 3 — Attaque de vol de session TCP sur telnet

Q 3.1 Rappeler le principe de l'attaque de vol de session TCP.

Correction

Insérer des données TCP dans une connexion TCP valide suite à la capture d'un paquet TCP. ◆

Q 3.2 Peut-on utiliser cette attaque sur une session entre un client et un serveur SSH communiquant sur IP ?

Correction

Non, car l'attaquant aurait besoin de la clé de chiffrement pour insérer des données SSH valides. ◆

Le protocole telnet est un protocole applicatif utilisé majoritairement pour l'exécution de commandes à distance sur des serveurs telnet. Cependant, contrairement à SSH, les messages telnet ne sont pas chiffrés, ce qui explique qu'il n'est plus guère utilisé. Le port par défaut utilisé par telnet est le port 23.

On imagine dans la suite de l'exercice un client (1.2.3.4:12345) et un serveur telnet (2.3.4.5:23). Le client a déjà ouvert sa session. L'attaquant capture le paquet ci-dessous.

`1.2.3.4:12345 -> 2.3.4.5:23 : Flags=A, SEQ=45243, ACK=85763, Len=200`

Q 3.3 Quel paquet l'attaquant peut-il envoyer s'il veut faire exécuter la commande `touch fichier` au serveur ?

Correction

`1.2.3.4:12345 -> 2.3.4.5:23 : Flags=A, SEQ=45443, ACK=85763, Len=17`

Le paquet TCP contiendra les 17 octets données TCP suivantes : `\rtouch fichier\r`. Le premier `\r` permet, si nécessaire, de terminer la commande précédente. Le deuxième valide la commande `touch`. ◆

Q 3.4 Si l'attaque réussit, la connexion TCP sera ensuite avortée soit par le client, soit par le serveur (ce qui en fait une des raisons qui font que l'attaque n'est pas très discrète). En effet, la connexion sera alors dans un état incohérent. Pourquoi ? Quels paquets seront ensuite échangés entre le client et le serveur ?

Correction

Si l'attaque réussit, le prochain paquet de données (légitime) envoyé par le client aura toujours le numéro `SEQ=45443` alors que le serveur attend maintenant l'octet `SEQ=45460` (suite à l'envoi de 17 octets par l'attaquant). Le serveur enverra donc des acquittements invalides et donc ignorés par le client. De son côté, le serveur recevra du client des octets de donneés qu'il considèrera comme des duplicats et qui seront donc aussi ignorés. Au bout d'un moment, une des deux parties mettra fin à la connexion avec un paquet RST. ◆

Exercice 4 — Scan Idle de nmap

Le scan Idle de l'outil nmap permet de tester l'état d'un port TCP (c'est-à-dire de déterminer si un processus écoute sur ce port). Il se base sur les aspects suivants des protocoles TCP et IP :

— Si un hôte reçoit un paquet SYN+ACK non attendu il répond par un RST.
— Un paquet RST non attendu est ignoré.
— Tout paquet IP contient un numéro d'identification (ID). Sur certains OS, ce numéro est simplement incrémenté de un après chaque envoi.

Le scan Idle fait intervenir trois hôtes :

— un attaquant (celui qui lance nmap) ;
— un zombie ;
— et une victime (celui qui est scanné par l'attaquant).

L'attaquant procède en trois étapes :

1. envoi d'un paquet SYN+ACK au zombie pour connaître son ID actuel
2. envoi d'un paquet SYN à la victime en usurpant l'IP du zombie
3. envoi d'un paquet SYN+ACK au zombie pour connaître son ID actuel

Q 4.1 Faire deux chronogrammes faisant apparaître les différents paquets échangés lors d'un scan (un premier chronogramme pour le cas où le port scanné est ouvert et un deuxième pour le cas où il est fermé).

Q 4.2 Dans quel cas est ce que l'attaquant peut conclure que le port est ouvert ? et que le port est fermé ?

Q 4.3 Quel est l'avantage de ce type de scan ?

Exercice 5 — Attaque TCP SYN flood

Q 5.1 Rappeler le principe de l'attaque TCP SYN flood.

Correction

Envoyer en rafale des paquets SYN sur un serveur cible (IP + port) pour saturer la mémoire allouée aux connexions mi-ouvertes. Une fois cette mémoire saturée, le serveur répondra négativement (avec un RST) aux demandes de connexion légitimes. ◆

Q 5.2 Quelle(s) adresse(s) IP et port(s) source l'attaquant doit-il utiliser dans les paquets forgés et envoyés pour attaquer un serveur ? Justifier.

Correction

Il ne faut pas que l'attaquant soit identifié (⇒ il ne doit pas utiliser son IP). L'attaquant doit donc usurper l'IP d'autres machines. De plus, il ne faut pas que l'adresse usurpée réponde par un RST au paquet SYN+ACK du serveur attaqué (sinon la connexion sort de la file des connexion mi-ouvertes).

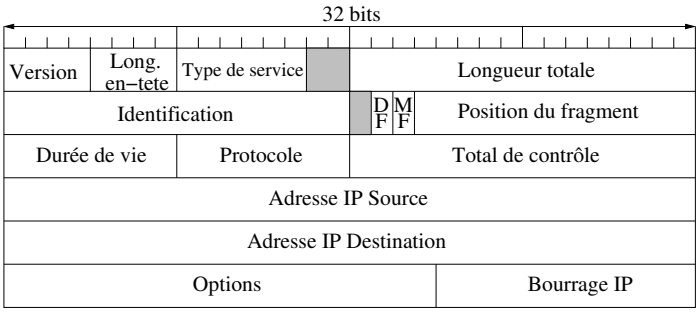
En pratique on peut utiliser des adresses IP et ports choisis aléatoirement. ◆

Structure des trames Ethernet

DA	SA	DL/EType	Données Ethernet + Bourrage Ethernet
6 o.	6 o.	2 o.	46–1500 o.

— DA = adresse destination
— SA = adresse source
— DL/EType = longueur du champ de données (DL) ou identifiant du protocole encapsulé (Etype).
Codes Etype (en hexadécimal) : IP = `08 00`, ARP = `08 06`, ...

En-tête IP



En-tête TCP