TP Sécurité réseau - Insecurity

***

Ayitic – Port au Prince
11 – 16 Août 2014

***

Instructeur: LOISEAU Lucien

Objectifs du TP

L'objectif de ce TP est de comprendre que si le réseau est essentiel pour atteindre facilement un certain nombre de ressources distantes, il est également très aisé de se rendre vulnérable. Ainsi, durant ce TP, nous allons nous familiariser avec les protocoles de l'Internet (`TCP/IP`, `DNS`, `HTTP`) et avec quelques outils d'audit réseau (`nmap`, `wireshark`, `arpspoof`, `sslstrip`) sous l'angle d'un attaquant afin de comprendre les enjeux de la sécurité. Durant ce TP nous allons donc apprendre à :

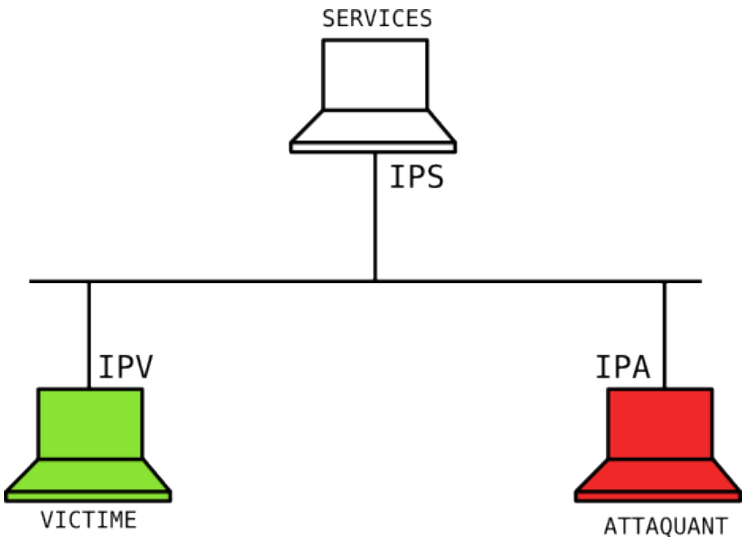
- Découvrir les hôtes et services présent sur un réseau local
- Détourner une connexion IP
- Écouter des informations circulant sur le réseau (images, text)
- Voler des identifications
- Déjouer une connexion sécurisée

Ce TP sera suivi par un autre TP durant lequel nous apprendrons à mettre en place des mécanismes de sécurité afin de pallier aux faiblesses du réseau.

Préparation du Banc

Identification du Matériel

Chaque banc est constitué de deux PCs, l'un sera considéré comme la `victime` et l'autre comme `l'attaquant`. L'ordinateur de l'instructeur étant lui considéré comme le `serveur distant` qui héberge différents services qui seront utilisés durant ce TP. Nous avons donc le schéma suivant :



Exemple de configuration d'un banc

Durant ce TP, nous utiliserons les termes suivants :

- **IPV** : Adresse IP de la victime mais nous utiliserons plus tard le nom d'hôte `victime.tp`
- **IPA** : Adresse IP de l'attaquant mais nous utiliserons plus tard le nom d'hôte `attaquant.tp`
- **IPS** : Adresse IP du serveur mais nous utiliserons plus tard le nom d'hôte `ayitic.tp`

De plus, nous serons amenés à taper des commandes sur l'ordinateur de la `victime` ainsi que sur celle de l'`attaquant`. Afin de ne pas s'emmêler les pinceaux, Chaque fois que des commandes seront tapés sur l'ordinateur de la `victime`, le fond sera vert tandis que les commandes tapées sur l'ordinateur de l'`attaquant` seront précisées avec un fond orange.

Scénario

Le scénario est le suivant : La `victime` va se connecter à différents services sur le `serveur distant` tels que `HTTP`, `DNS` ou `FTP`. Le réseau de la salle sera donc vu comme une abstraction du réseau

Internet. Nous allons voir à travers ce TP comment un `attaquant` peut s'y prendre pour attaquer la victime afin d'intercepter son trafic, de le modifier, de le forger ou bien même de rendre indisponible l'accès au service.

Installation des outils

Assurez vous que les outils suivants soient bien installés sur le système de l'`attaquant` :

- `nmap`
- `wireshark`
- `arpspoof`
- `sslstrip`

Vous pouvez vérifier cela en essayant ces commandes dans un terminal. Si un de ces outils est manquant, appelez l'instructeur. La plupart des commandes nécessiteront les droits root. Pour les obtenir tapez sudo -s suivi du mot de passe (ayitic), si cela ne fonctionne pas appelez l'instructeur !

```
$ whoami
victime
$ sudo -s
# whoami
root
```

Découverte du réseau

La découverte du réseau consiste à dessiner une carte du réseau en identifiant les machines, le plan d'adressage et les services. L'outil incontournable pour ce type d'opération se nomme `nmap`. Nmap est un scanner de réseau qui permet de détecter des hôtes et des services sur un réseau. Avant de nous lancer dans la découverte du réseau, nous allons commencer par récupérer quelques informations sur la configuration locale du PC de l'attaquant.

Configuration des interfaces locales

Nous allons commencer par récupérer la configuration des interfaces locales à l'aide de la commande `ifconfig` :

```
$ ifconfig
```

Question 1 : Quelle interface est utilisée pour communiquer avec le réseau de la salle et quelle est son adresse IP ?

Question 2 : Quel le masque du sous-réseau local ? Combien d'adresse IPs sont adressable dans cette plage d'IP ?

Question 3 : À quoi sert l'interface lo ?

Répétez l'opération sur l'ordinateur de la victime et déterminez `IPA` et `IPV`.

Important : Une fois que vous avez trouvé l'adresse de la victime IPV, ajoutez la en root dans /etc/hosts à la fois sur l'ordinateur de la victime et sur celle du client :

```
# echo "IPV victime.tp" >> /etc/hosts
```

Configuration des routes

Nous allons ensuite nous renseigner sur la configuration de la table de routage, celle-ci nous indiquera par quelle interface sortent les paquets. Tapez la commande suivante afin d'afficher la table de routage du système :

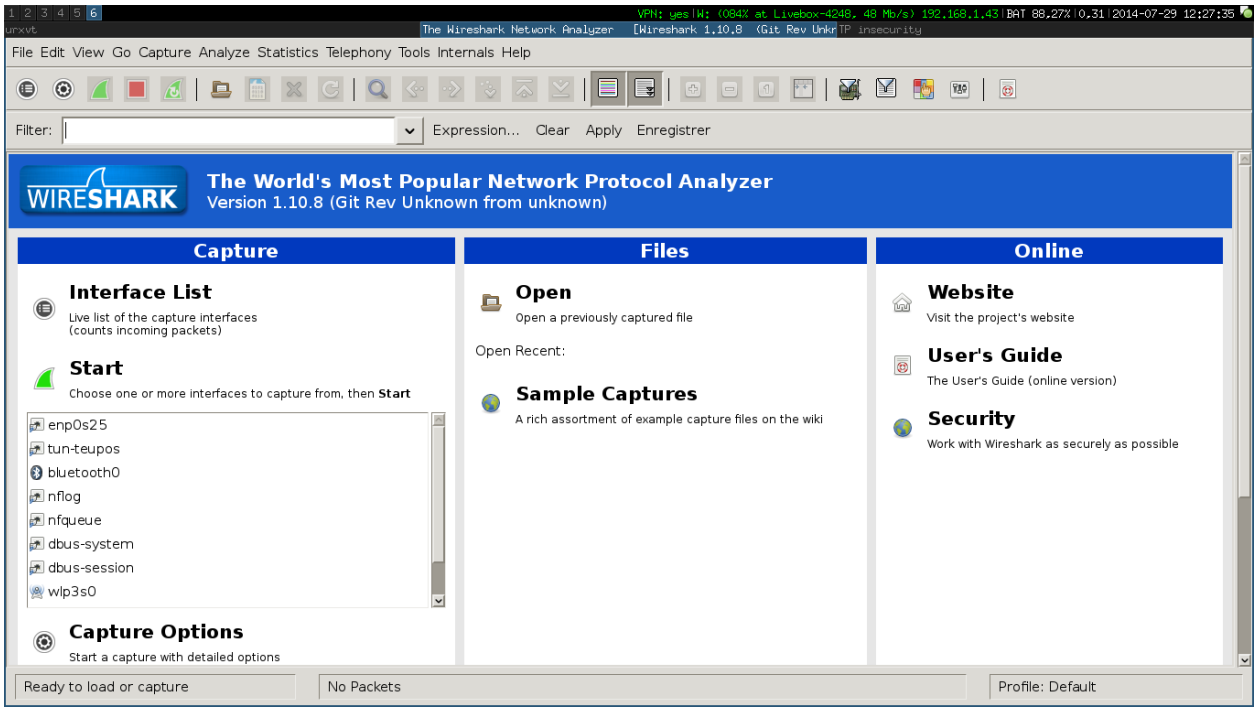
```
$ ip route show
```

Question 4 : Quelle route est utilisée pour router les paquets à destination du réseau local ?

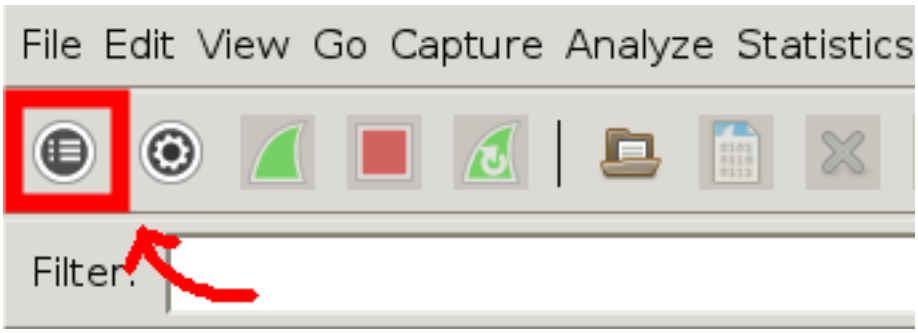
Question 5 : À quoi sert la route par défaut ?

Introduction à Wireshark et NMAP

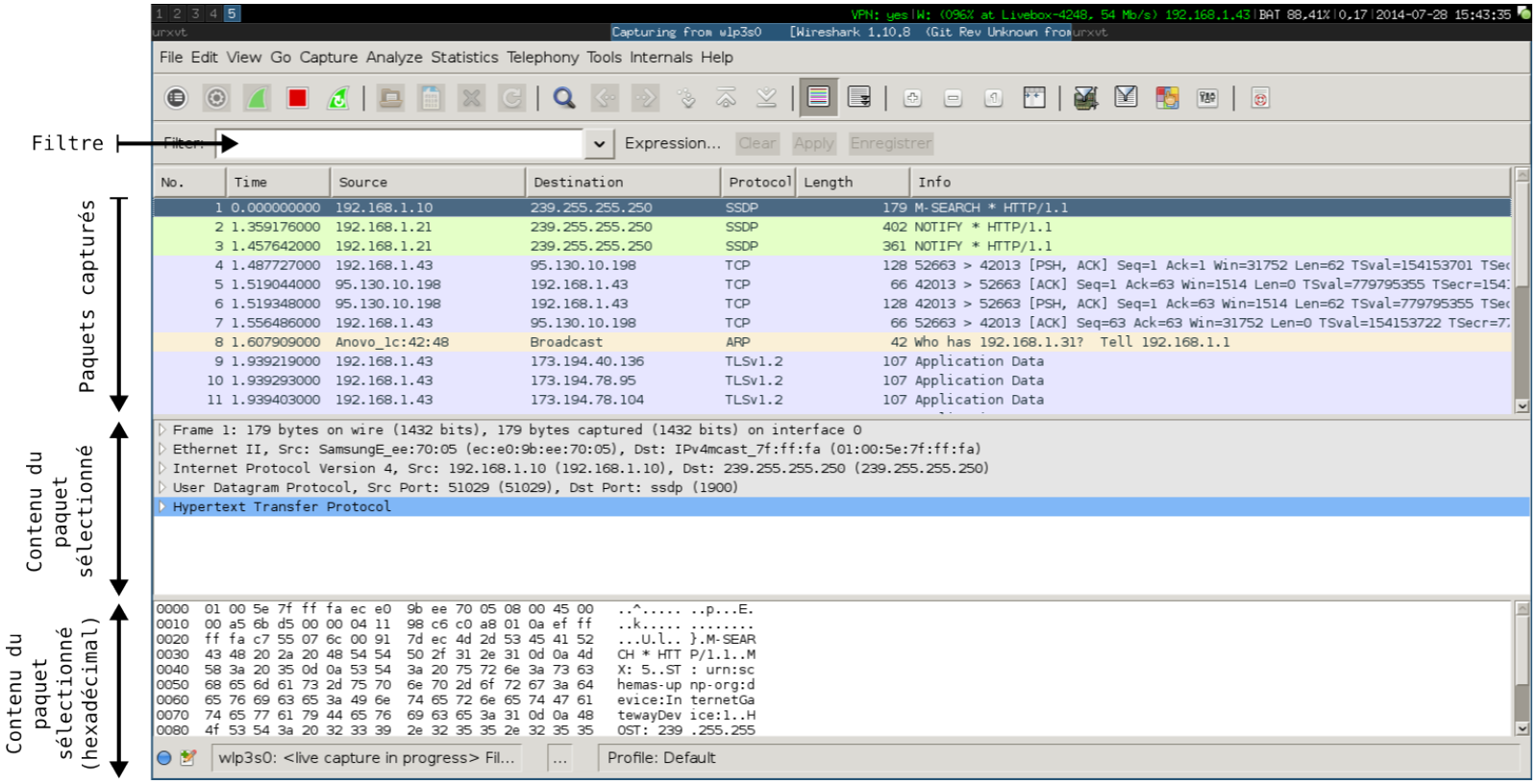
Nous allons commencer par utiliser `nmap` pour découvrir toutes les machines connectées au sous-réseau local. Nous allons dans le même temps démarrer wireshark pour voir comment fonctionne `nmap`. Démarrez wireshark en root et commencer la capture sur l'interface identifiée à la question 1. Vous devriez avoir quelque chose qui ressemble à ceci :



Wireshark est un outil permettant de visualiser les paquets qui parte ou arrive sur la machine. Pour démarrer une trace, cliquez sur le bouton situé en haut à gauche du logiciel :



Cela va ouvrir un popup vous permettant de sélectionner l'interface à écouter. À l'aide de la question 1, lancez une trace sur la bonne interface. Des paquets vont apparaître dans la fenêtre principal comme sur la figure suivante :



Il peut y avoir un grand nombre de paquet, afin de s'y retrouver vous pouvez utiliser le filtre pour réduire la liste. Il est possible de filtrer en fonction du protocole par exemple `ip`, `icmp`, `arp`, `tcp`, `ip`, `http`, `ftp`. Pour chacun des protocoles il est possible de filtrer plus précisément, par exemple pour filtrer en fonction de l'adresse IP :

- `ip.src == 192.168.1.12` : n'affiche que les paquets dont l'IP source est `192.168.1.12`

- `ip.dst == 192.168.1.1` : n'affiche que les paquets dont l'IP destination est `192.168.1.1`
- `ip.addr == 192.168.1.1` : n'affiche que les paquets dont la source **OU** la destination est `192.168.1.1`

De même pour TCP on peut filtrer en fonction du port :

- `tcp.src == 80` : n'affiche que les paquets dont le port `TCP` source est `80` (HTTP)
- `tcp.dst == 80` : n'affiche que les paquets dont le port `TCP` destination est `80`
- `tcp.addr == 80` : n'affiche que les paquets dont le port `TCP` source **OU** destination est `80`

Le contenu du paquet est décapsulé dans la troisième fenêtre où on peut ainsi décortiquer le paquet. Faites quelques essais pour vous faire la main avec wireshark. Faites un `ping` sur une machine quelconque et regardez les paquets avec wireshark

Question 6 : Quel est le nom du protocole utilisé pour effectuer un ping ? Comment se déroule un ping ?

I. NMAP : Découverte d'hôtes

`nmap` peut détecter les hôtes connectés avec l'option `-sP`. Pour `nmap`, tout ce qui n'est pas une option est une cible. Une cible peut être une IP fixe (*ex :* `192.168.0.1`), une plage d'adresse IP (*ex* `192.168.0.0/24`), un nom de domaine (*ex :* `google.fr`) ou même un fichier contenant des cibles (une cible par ligne). Par exemple si la plage à scanner pour détecter les hôtes connectés est `192.168.0.0/24` on lancerait nmap de la façon suivante :

```
$ nmap -sP 192.168.0.0/24
```

À l'aide de la réponse à la question 2, effectuez un scan du sous-réseau local.

Question 7 : En regardant les traces fournis par wireshark, expliquez le fonctionnement de ***nmap -sP***

Question 8 : À quoi servent les requêtes ARP ?

Question 9 : Combien de machines sont présentes sur le sous-réseau local ?

II. NMAP : Découverte de service

On peut également demander à nmap de nous scanner les services disponibles (`TCP` ou `UDP`) pour un hôte en particulier ou pour une plage d'hôte. Il y a différentes façon de savoir si un service est disponible, la plus simple étant d'essayer de s'y connecter et de voir ce que nous répond. Pour un scan TCP on utilisera le paramètre `-sS` et pour un scan UDP on utilisera le paramètre `-sU`. Choisissez une adresse IP parmis la liste précédente et procédez à une détection de service TCP, par exemple :

```
# nmap -sS 192.168.0.15
```

Question 10 : En regardant les traces fournis par wireshark, expliquez comment nmap découvre qu'un port TCP est ouvert ? Qu'un port TCP est fermé ?

Le scan UDP peut être particulièrement long car contrairement à TCP, lorsqu'un port est fermé il n'envoie généralement aucune réponse et il faut attendre que nmap timeout. Aussi ne vous inquiétez pas si le scan UDP n'aboutit pas.

```
# nmap -sU 192.168.0.15
```

III. NMAP : Fingerprinting d'OS

En rajoutant un paramètre, on peut demander à `nmap` de faire de la détection d'OS. Cette opération prend un peu plus de temps car pour cela nmap va étudier le comportement de la machine ciblé face à différentes requêtes (par exemple une connexion `TCP`). En effet, les OS ne réagissent pas toujours de la même façon et il est possible de caractériser un OS en suivant son comportement, cela s'appelle du `fingerprint` d'OS. L'option pour faire une découverte d'OS est `-O`. Choisissez une adresse IP parmis la liste précédente et procédez à une détection d'OS :

```
# nmap -O 192.168.0.15
```

La détection d'OS se base sur une base de donnée de comportement d'OS connu et n'est donc pas une science exact. Il est possible que la détection échoue, si c'est le cas recommencez avec une autre adresse IP.

IV Exercice

*Question 11 : Utilisez nmap pour trouver l'adresse l'adresse **IPS** de la machine de l'instructeur sachant qu'elle héberge (au moins) un serveur HTTP (port TCP/80) et HTTPS (port TCP/443).*

Important : Une fois que vous avez trouvé l'adresse du serveur, ajoutez le en root dans /etc/hosts à la fois sur l'ordinateur de la victime et sur celle du client :

```
# echo "IPS ayitic.tp" >> /etc/hosts
```

```
# echo "IPS ayitic.tp" >> /etc/hosts
```

Le fichier `/etc/hosts` est un fichier qui contient des homs d'hôtes associés à des adresses IP, par exemple localhost pour `127.0.0.1` dorénavant le serveur accessible à l'adresse `IPS` sera également accessible avec le nom d'hôte `ayitic.tp`

Avec la machine de la `victime`, utilisez le navigateur (Firefox) pour vous connecter à **ayitic.tp**, si c'est la bonne IP, vous devriez vous connecter à une page nommée « Ayitic 2014 ». Le site web de la machine de l'instructeur héberge un site web accessible à l'adresse suivante :

Vous devriez voir apparaître le portail Bleu « Ayitic - 2014 », si ce n'est pas le cas, appelez l'instructeur.

Man-In-The-Middle

L'attaque **Man-In-The-Middle** (MITM) consiste pour une machine malveillante à se positionner sur le chemin des paquets transitant entre une victime et un serveur distant. Cette position permet à un attaquant malveillant d'intercepter des communications, de les modifier à la volée ou même de les supprimer.

Question 12 : Si le serveur distant était un vrai site web sur Internet, qui se trouve normalement sur le chemin entre l'utilisateur et le site web ?

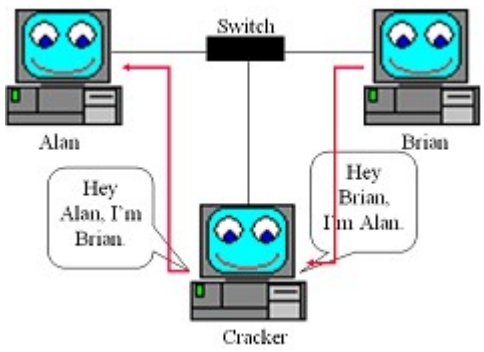
Question 13 : Pensez vous que nous pouvons leurs faire confiance, pourquoi ?

ARP Cache Poisoning : `arpspoof`

Si l'attaquant ne se trouve pas naturellement sur le chemin, il est cependant possible sur un réseau local de forcer à être un intermédiaire grâce à l'attaque dite de « `ARP cache poisonning` ». Cette technique consiste à corrompre le cache **ARP** de deux machines A et B du chemin pour :

- Faire passer l'attaquant pour A auprès de B
- Faire passer l'attaquant pour B auprès de A

Le schéma suivant résume l'attaque :



Dans notre scénario, Alan est la victime et Brian est le serveur, Cracker est bien sur l'attaquant. Pour mettre en œuvre notre attaqye on va utiliser la commande `arpspoof` disponible sous Linux dans le paquet `dsniff` :

```
# arpspoof -h
Version: 2.4
Usage: arpspoof [-i interface] [-t target] host
```

Ce programme va permettre de corrompre le cache ARP de `target` en lui faisant croire que l'adresse MAC de l'`host` est celle de l'`attaquant`. On va donc corrompre deux personnes :

- La `victime` où l'on se fera passer pour le `serveur`
- Le `serveur` où l'on se fera passer pour la `victime`.

Ouvrez un terminal et lancez la commande suivante :

```
# arpspoof -i eth0 -t IPV -r IPS
```

ATTENTION : Laissez tourner `arpspoof` durant tout le TP jusqu'à ce qu'il vous soit demandé de l'éteindre !

*Question 15 : À l'aide de wireshark (filtre **arp**), expliquez comment fonctionne arpspoof*

Si on essaye de se connecter au site web depuis la machine de la victime cela ne fonctionne pas. Sur l'ordinateur de l'attaquant, tapez la commande suivante en `root` :

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Question 16 : Que fait cette commande ? Est-ce que la victime peut se connecter au site maintenant ?

Appelez l'instructeur pour qu'il valide cette étape !

Étude du protocole HTTP

I. Échauffement

L'attaquant se trouve maintenant sur le chemin entre la victime et le serveur, c'est à dire que c'est un « intermédiaire » qui relais les paquets de la victime à destination du serveur et inversement (*Si ce n'est pas le cas appelez l'instructeur*). Il est donc possible pour l'attaquant « d'écouter » les paquets qui transitent par son ordinateur. Nous allons maintenant nous intéresser au protocole HTTP. Lancez une nouvelle trace sur l'ordinateur de l'`attaquant` avec wireshark et filtrer en fonction de l'adresse **IPV** de la victime : (Filtre : `ip.addr == IPV`)

ATTENTION: suivez les instructions, ne chargez les pages web qu'une seule fois et ne cliquez pas sur les liens si cela n'est pas demandé. Certaines questions nécessitent que vous fassiez les choses dans l'ordre !

Sur l'ordinateur de la `victime` connectez vous avec le navigateur sur la page de test et observez les traces obtenues sur l'ordinateur de l'`attaquant` :

```
http://ayitic.tp/test/
```

Du fait que l'ordinateur de l'attaquant relais les trames, chaque paquet apparaît en double ce qui rend la lecture difficile. Utilisez le filtre suivant :

```
ip.addr == IPV &&!icmp &&!(frame.coloring_rule.name == « Bad TCP »)
```

Explication :

- on ne veut que les paquets étant à destination ou depuis la victime (ip.addr == IPV)
- on filtre les paquets icmp (!icmp)
- on ne veut pas des erreurs TCP (dû au relayage wireshark croit que ce sont des erreurs)

Question 17 : Comment s'établit la connexion TCP (avant le premier paquet HTTP) ? Quel est le port TCP destination ? Quel est le port TCP source ? (répondez à la question sous la forme d'un chronogramme)

Question 18 : Expliquez à quoi servent les paquets TCP [ACK] ?

Afin d'étudier le protocole HTTP, nous allons rendre les traces encoreun peu plus lisible en ne

sélectionnant que le protocole http. Utilisez le filtre suivant :

```
http && ip.addr == IPV && !icmp && !tcp.analysis.retransmission
```

Question 19 : Combien de requêtes GET ont été effectuées en tout ? À quoi correspondent elles ?

Vérifiez la réponse à cette question en regardant le code source de la page sur l'ordinateur de la `victime` (Ctrl+U dans le navigateur Firefox)

II. On entre dans les détails :-]

Étudiez la section relative au protocole HTTP dans les réponses aux requêtes.

Question 20 : Que contiennent ces réponses HTTP ?

Question 21 : Quelle est la version du protocole HTTP utilisée pour effectuer les requêtes ?

Question 22 : Expliquez d'où vient la requête « GET body.png »

Question 23 : Quelle est la requête `GET /favicon.ico` *et quelle est la réponse du serveur HTTP ? Pourquoi ?*

Notez la valeur de l'option `Etag` dans la réponse du serveur à la requête « GET style.css » et Rechargez la page de test avec « Ctrl+R »

Question 24 : Regardez la valeur de l'option `If-None-Match` *dans cette deuxième requête de style.css et expliquez pourquoi le serveur n'a pas renvoyé les données de style.css dans sa réponse ?*

Depuis l'ordinateur de la victime cliquez sur le lien « `Lien vers la page de test 2` » et regardez les traces wireshark sur l'ordinateur de l'`attaquant` :

Question 25 : À quoi correspond le mot clé `Referer` *dans l'entête HTTP des requêtes GET ?*

Question 26 : À quoi correspond le mot clé `User-Agent` *dans l'entête HTTP des requêtes GET ?*

À savoir

L'objectif de cet échauffement est de vous montrer que lorsqu'on accède à une ressource sur Internet, on donne tout un tas d'information appelées metadata. Dans cet exemple, un serveur peu soucieux de la vie privée des utilisateurs (google par exemple) ou quelqu'un se trouvant sur le chemin (votre fournisseur d'accès par exemple) peut, simplement en regardant les entêtes TCP/IP et HTTP :

– Connaître votre adresse IP source (et donc votre localisation géographique)
– Connaître l'IP de destination ainsi que le service (port tcp)
– Connaître la ressource que vous souhaitez récupérer avec la requête GET
– Connaître de quel site vous provenez grâce à l'option REFERER
– Connaître La version de votre navigateur et du système d'exploitation avec l'option USER-AGENT
– Vous tracker grâce à l'option ETAG (normalement il est dépendant de la resource mais rien n'empêche le serveur web de le rendre unique par utilisateur), et tout un tas d'autre option vous identifiant de façon unique (cookie de session, plugins du navigateur, langues, encodage, etc.)

Il faut donc être conscient qu'on laisse beaucoup de traces !

III. Interception de données

Sur l'ordinateur de l'`attaquant` faites une nouvelle trace wireshark en cliquant sur le bouton vert avec une flèche « Restart the running live capture ». Cette fois-ci, la victime va se connecter à une page « perso » contenant des photos qui sont censées être privées. Connectez vous avec l'ordinateur de la `victime` sur la page suivante :

`http://ayitic.tp/privee/`

Observez les traces wireshark en filtrant sur le protocole `http`. Cherchez dans les traces les requêtes `HTTP GET` effectuées par le navigateur de la victime pour les images « `chat.jpg` », « `lapin.jpg` » et « `hibou.jpg` ». Étudiez les réponses du serveur à ces requêtes en fouillant dans les entêtes des réponses.

Question 27 : Quelle est la taille du fichier chat.jpg? En quoi cela est il exprimé ?

Il est possible pour l'attaquant d'afficher l'image contenu dans la réponse HTTP grâce à wireshark. Sélectionnez la réponse HTTP contenant une image, et faites un clic droit sur la section « `JPEG File Interchange Format` » et sélectionnez « `Export Selected Packet Bytes` ». Enregistrez cela sur le Bureau sous le nom « `image1.jpg` ». Ouvrez maintenant cette image depuis votre bureau et celle-ci devrait apparaître !

À savoir

À ce stade vous devriez vous rendre compte qu'il est facilement possible de cette manière d'intercepter n'importe quelle donnée (image, son, vidéo, texte) transitant en clair sur le réseau !

IV. Interception de mot de passe

Nous allons maintenant mettre en évidence qu'il est aussi simple d'intercepter un mot de passe que d'intercepter une image. **Démarrez une nouvelle trace wireshark** sur l'ordinateur de l'`attaquant` et avec le navigateur de la `victime` connectez vous à l'adresse suivante :

`http://ayitic.tp/banque/`

Question 28 : Dans l'entête de la première réponse, à quoi sert d'après vous l'option `set-cookie` *?*

Il s'agit d'un site simulant un accès client à un site bancaire. Vous devriez voir apparaître un formulaire vous demandant de remplir votre login et mot de passe. si vous êtes le banc numéro 1, le login et le mot de passe seront `banc01`. Si vous êtes le banc numéro 2, ce sera `banc02` etc. Si vous n'avez pas de numéro de banc, appelez l'instructeur.

Sur l'ordinateur de la victime, connectez vous à l'aide de vos identifiants et observez les traces wireshark sur l'ordinateur de l'`attaquant` et en particulier la requête `POST` obtenue :

Question 29 : Comment les mots de passes ont-ils été envoyés au serveur ?sont ils en clair par défaut ?

Depuis l'ordinateur de la victime, effectuez un virement quelconque à l'aide du formulaire disponible sur le site. Étudiez la nouvelle requête GET obtenue.

Question 30 : Quelle ressource (page) est appelée par le navigateur pour effectuer le virement ?

Question 31 : Comment ont été passé les paramètres (`accountid` *et* `montant`*) ? Expliquez la différence entre le passage de paramètre avec GET et celui avec POST.*

V. IP Spoofing : Usurpation d'adresse IP

Question 32 : Bien que cela soit tentant, il serait fort imprudent de la part de notre attaquant de se connecter directement au site web avec les mots de passe qu'il vient de voler, pourquoi ?

L'`attaquant` ne veut pas que son adresse IP soit enregistrée dans les logs du serveur. En revanche comme il se trouve sur le chemin entre la victime et l'attaquant, il peut tout simplement bloquer les messages en provenance de la `victime` et usurper son adresse IP. Il pourra ensuite se connecter sur le serveur avec sa nouvelle adresse IP et effectuer le virement qui lui convient.

Pour cela il faut d'abord arrêter de relayer les messages entre la `victime` et le serveur :

```
# echo 0 > /proc/sys/net/ipv4/ip_forward
```

essayez de vous connecter au site depuis l'ordinateur de la `victime`, normalement cela ne fonctionne plus car l'attaquant bloque les messages en provenance de la `victime`.

Il faut maintenant configurer l'`attaquant` avec sa nouvelle adresse IP, cela peut être fait très simplement avec la commande `ip`, on commence d'abord par lui supprimer l'adresse `IPA` qu'il possède pour l'instant :

```
# ip addr del IPA/24 dev eth0
```

Et on ajoute ensuite l'adresse `IPV` de la `victime` :

```
# ip addr add IPV/24 dev eth0
```

Sur l'ordinateur de l'`attaquant`, connectez vous maintenant sur le site de la banque et effectuez un virement quelconque. Regardez dans l'historique des virements que c'est bien l'adresse IP de la victime qui a été enregistré.

Question 33 : Pensez vous qu'il soit possible pour le serveur de détecter cette attaque ?

Nous allons essayer de nous rendre encore plus invisible en faisant une nouvelle requête HTTP mais en forgeant nous même l'entête. Cela est possible de plusieurs manières, la plus simple étant d'écrire notre requête HTTP dans un fichier et de l'envoyer au serveur avec `netcat` ! En effet, le protocole HTTP étant textuel, il est extrêmement simple de forger nous même ce paquet. Exemple : Tapez ce qui suit

dans un fichier que l'on appelera `request.txt` :

```
~ fichier : request.txt ~
01GET / HTTP/1.1
02Host: ayitic.tp
03
04
```

Il s'agit d'une requête `HTTP GET` toute simple qui demande simplement la ressource `/` (le portail ayitic 2014 qu'on a vu tout à l'heure). Il n'y a qu'une seule option Host qui précise l'adresse IP du serveur (cette option est la minimum requise). **Attention** : La ligne 03 doit également contenir un retour charriot. Tapez la commande suivante pour l'envoyer au serveur :

```
# cat request.txt | sed -e "s/$/\r/" | netcat IPS 80
```

Nous venons de reproduire à la main ce que fait le navigateur Firefox, Explications :

- la commande `cat request | sed -e "s/$/\r/"` permet de lire le fichier (avec `cat`) et d'envoyer la sortie à `sed` qui va ajouter le caractère `\r` à chaque fin de ligne de façon à ce que chaque ligne se termine par `\n\r` (Carriage Return Line Feed)
- la commande `netcat` qui se connecte au serveur `IPS` sur le port 80 et qui lui envoie ce qu'il reçoit de l'entrée standart (c'est à dire la requête contenu dans le fichier modifié avec `sed`).

Si cela a fonctionné, vous devriez voir apparaître la réponse suivi de la page Ayitic 2014 directement dans le terminal !

Question 34 : écrivez une requête permettant de faire un GET sur le site de la banque.

Vous devriez recevoir le code HTML de la page de connexion du site de la banque. Nul besoin de faire une requête POST, il nous suffit d'utiliser un cookie de session valide, celui de la victime par exemple ;-)

Question 35 : Modifiez votre requête de façon à inclure un cookie de session valide (Regardez les traces wireshark précédente pour vous aider)

Vous devriez maintenant recevoir le code HTML de la page du compte utilisateur, c'est à dire contenant les historiques de virement etc. L'attaquant souhaiterait maintenant effectuer un virement de 10 000 € vers son compte bancaire 1234567890.

Question 36 : Modifiez la requête précédente de façon à effectuer ce virement (voir question 29 et 30). Profitez en pour modifier le User-Agent pour qu'il ressemble à celui de la victime

Appelez l'instructeur pour qu'il valide votre hacking :-) !

Nous allons maintenant nous remettre dans une condition de Man-In-The-Middle classique sans `IP-spoofing`, tapez les commandes suivantes :

```
# ip addr del IPV/24 dev eth0
# ip addr add IPA/24 dev eth0
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Faites une nouvelle trace wireshark et connectez vous depuis l'ordinateur de la victime sur le site de la banque pour vérifier que cela fonctionne comme avant.

Déjouer le HTTPS

Tout cela a été possible car toutes les communications entre l'ordinateur de la victime et le serveur ont été effectuées en clair. Un site bancaire sérieux prendrait bien sur soin de mettre en place une connexion sécurisée au site en utilisant le SSL. Nous allons voir dans les exercices suivants que cela n'est pas toujours suffisant si la `victime` fait preuve d'imprudence.

I. Connexion au site sécurisé

Démarrez une nouvelle trace avec wireshark sur l'ordinateur de l'attaquant. Modifiez le filtre pour ajouter le `ssl` avec « `||  ssl` » de façon à obtenir le filtre suivant :

```
(http ||ssl) && ip.addr == IPV && !icmp && !tcp.analysis.retransmission
```

Maintenant connectez vous avec le navigateur de la victime à la version sécurisée du site bancaire :

```
http://ayitic.tp/secure
```

Vous devez normalement obtenir ceci qui vous invite à cliquer sur le cadenas pour vous connecter via le formulaire de connexion sécurisé :

Observez le code source de cette page soit sur l'ordinateur de la victime (Ctrl+U), soit dans wireshark En regardant dans la réponse au premier GET dans la section « Line-Based Text Data: text/html ».

Question 37 : Sans appuyer sur le bouton, expliquez en quoi consiste la sécurisation ?

Appuyez sur le bouton de connexion au site sécurisé et regardez les traces wireshark obtenue.

Question 38 : Sur quel port se fait une connexion en HTTPS ?

Question 39 : À partir de quelle couche protocolaire les informations sont elles chiffrées ?

Il apparaît difficile pour l'attaquant de lire les informations sécuriser avec TLS (Transport Layer Security) anciennement appelé SSL (Security Socket Layer). Dans le Prochain TP nous apprendrons à sécuriser un serveur HTTP mais pour l'instant concentrons nous sur le fait que les données sont chiffrés et donc illisible pour l'attaquant.

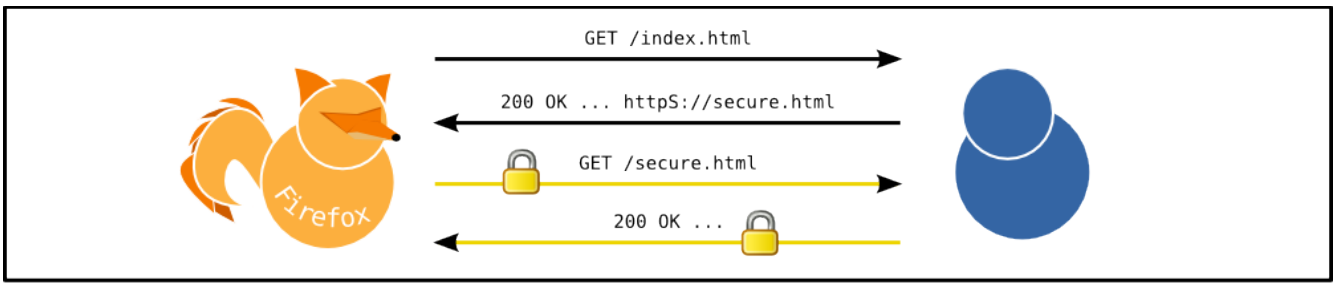
Sur l'ordinateur de la victime rentrez vos login et mot de passe et effectuez un virement quelconque. En regardant les traces correspondantes sur wireshark on s'aperçoit qu'il n'y a rien à faire, les données sont parfaitement bien chiffrées.

I. Casser une communication HTTP sécurisée avec SSL

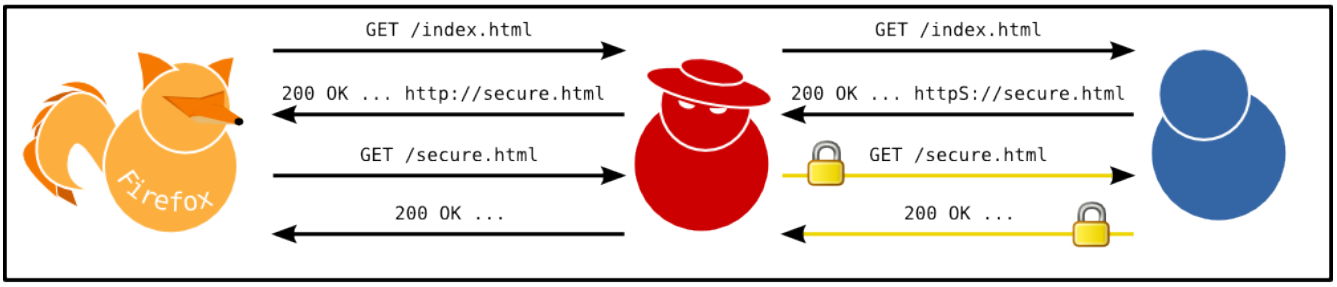
La faiblesse du httpS (http + SSL) réside dans le fait que pour accéder à une page chiffrée il faut généralement d'abord passer par une page non chiffrée. Par exemple, pour accéder à la version sécurisé du site (sur le port 443), il a fallut y être redirigé depuis la page en clair non sécurisé sur le port 80. En effet, rare sont les utilisateurs qui font l'effort de rajouter le 's' à la suite de http dans la barre d'adresse (une page sécurisée commence par https:// et se connecte sur le port 443, tandis qu'une page non sécurisée comment par http:// et se trouve sur le port 80)

Cela est exploitable par l'attaquant car ce qu'il peut faire, c'est modifier à la volée toutes les réponses du serveur pour enlever le 's' de chaque chaque lien hypertexte de type « https:// » avant de les relayer à la victime. L'attaquant communique donc de façon sécurisé avec le serveur mais la victime communique en clair avec l'attaquant !

Le schéma ci-dessous résume l'attaque :



Déroulement normal



Déroulement avec l'attaque SSLSTRIP

SSLStrip ne se contente pas d'enlever les 's' de https, il enregistre également tout les paramètres envoyés au sein d'un POST (normalement) sécurisé dans un fichier de log, et comble de l'ironie, si la victime fait une requête pour l'image « `favicon.ico` » l'`attaquant` lui envoie l'image d'un petit cadenas pour rassurer la victime.

Pour lancer l'attaque, il faut lancer le programme sslstrip sur l'ordinateur de l'attaquant :

```
# sslstrip -f
```

sslstrip fonctionne comme un petit serveur web qui écoute sur le port 10000. Il attend de recevoir des requêtes et les forward au « vrai » serveur quand il en reçoit et s'occupe ensuite de nettoyer les réponses du serveur avant de les relayer à la victime.

On va détourner le trafic de telle sorte que tout les paquets à destination du port 80 soit envoyés sur le port 10000. Cela est possible avec la commande suivante :

```
# iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 10000
```

Grâce à cette commande, toutes les communications entre la victime et le serveur sont relayés (comme avant) excepté celle à destination du port 80 qui sont redirigé sur l'ordinateur de l'attaquant sur le port 10000 (c'est à dire vers notre « proxy » sslstrip).

Depuis l'ordinateur de la victime connectez vous à la page d'accueil de la banque (en http) de façon à obtenir l'invitation à nous connecter au site sécurisé comme précédemment. Maintenant appuyez sur le bouton pour procéder à la connexion soit-disant sécurisée et connectez vous avec vos login et mot de passe.

Cela fonctionne et pourtant la connexion n'est plus du tout sécurisé ! Si la victime est attentive, elle s'apercevrait bien en regardant la barre d'adresse que la connexion n'est pas en httpS mais bien en clair en http ! Cette attaque sournoise est redoutablement efficace. Maintenant de retour sur l'ordinateur de l'attaquant, sslstrip a crée un fichier de log appelé sslstrip.log, ouvrez le et découvrez votre césame.

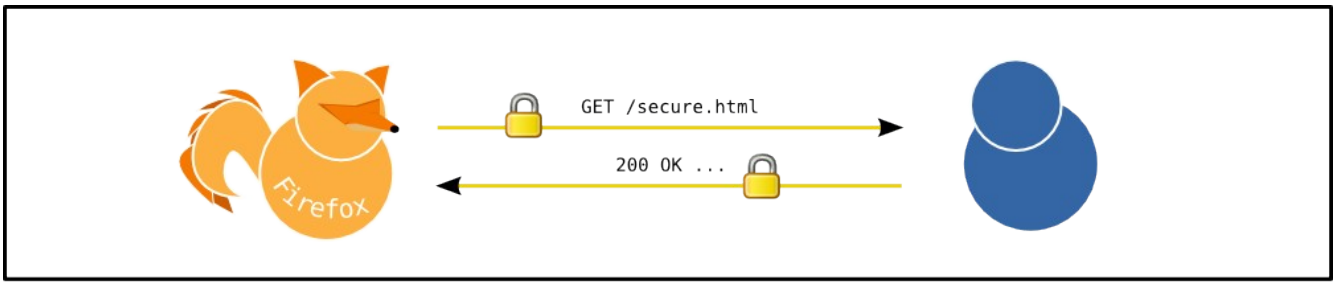
Félicitation, vous avez déjouez une connexion chiffrée en HTTPS !

On enlève la règle `Iptables` et on coupe `sslsniff` :

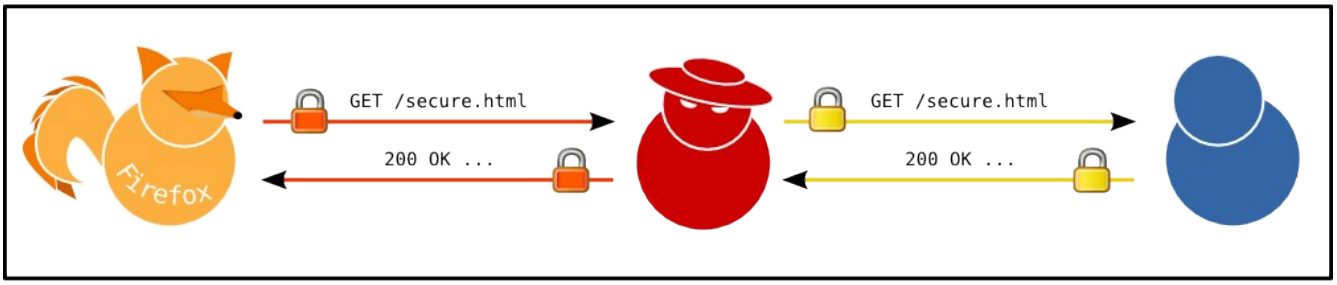
```
# iptables -t nat -D PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 10000
```

I. Tromper une communication HTTP sécurisée avec SSL

D'autres attaques sont possibles jouant sur le manque de rigueur de la victime comme par exemple avec le programme sslsniff. Dans cette attaque, l'attaquant va envoyer ses propres clés publiques de façon à couper la connexion en deux. La victime aura donc une communication sécurisé avec l'attaquant et l'attaquant aura lui même une communication sécurisée avec le serveur. Le schéma suivant résume l'attaque :



Déroulement normal



Déroulement avec l'attaque SSLSNIFF

Dans un terminal on commence par créer un certificat qui sera utilisé par l'attaquant qui sera utilisé par l'attaquant :

```
# openssl req -new -x509 -keyout server.pem -out server.pem -days 365 -nodes
Generating a 1024 bit RSA private key
..............................++++++
....++++++
writing new private key to 'server.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:HT
State or Province Name (full name) [Some-State]:Haiti
Locality Name (eg, city) []:Port-Au-Prince
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Certification
Authority of Haiti (CA) Level 3
Organizational Unit Name (eg, section) []:Security Administration of Commerce
Common Name (e.g. server FQDN or YOUR name) []:ayitic.tp
Email Address []:
```

Répondez aux questions de façon à tromper la victime avec des noms d'organisations sérieuses.

Maintenant lancez `sslsniff` avec la commande suivante :

```
# sslsniff -a -c server.pem -s 10000 -w sslsniff.log
```

Comme pour sslstrip, On va détourner le trafic de telle sorte que tout les paquets à destination du port 443 (HTTPS) soit envoyés sur le port 10000. :

```
# iptables -t nat -A PREROUTING -p tcp --destination-port 443 -j REDIRECT --to-port 10000
```

Depuis l'ordinateur de la victime connectez vous à la page d'accueil de la banque en HTTP**S**. Une alerte Firefox s'affiche pour dire que le certificat n'est pas valide, **et pour cause c'est celui de l'attaquant !**. Cependant bien souvent, les utilisateurs imprudents ne prennent pas le temps de lire les message d'erreur. Comme le fairait un utilisateur imprudent, cliquez sur :

`Add Exception → Confirm Security Exception`

Maintenant connectez vous sur le site avec vos identifiants et ouvrez le fichier sslstrip.log sur l'ordinateur de l'attaquant.

Encore une fois, nous venons de montrer que la sécurité n'est pas toujours automatisable !

Vous pouvez maintenant arrêter arpspoof, sslsniff et wireshark

Conclusion

La leçon de cet exercice est qu'il est extrêmement risqué de ne pas chiffrer les informations sur Internet car vous ne pouvez pas savoir qui se trouve entre vous et le serveur distant. Internet est un réseau d'interconnexion qui implique un grand nombre d'acteur dont certain sont au mieux peu soucieux de votre vie privée, ou au pire malveillant et souhaitant voler vos informations pour faire par exemple de l'espionnage industriel.

Nous avons étudié en détail le déroulement d'une attaque Man-In-The-Middle et avons constaté avec quelle facilité il était possible de voler des informations non-chiffrés ! Nous avons ensuite étudié le déroulement de deux attaques contre le HTTPS qui montre que dans la sécurité tout n'est pas automatisable et que bien souvent le maillon faible est l'humain. Ces deux dernières attaques n'auraient jamais pu aboutir si l'utilisateur sérieux avait pris soin de vérifier que la connexion était effectivement bien sécurisé et que les certificats étaient valides.

Dans le prochain TP, nous étudierons justement comment vérifier qu'une connexion est valide en tant qu'utilisateur et nous apprendront comment sécuriser un serveur WEB en tant qu'administrateur via l'architecture PKI (Public Key Infrastructure).