1 Objectifs

L'objectif de ce TP est de vous initier à certaines techniques dites de *cryptographie* afin de sécuriser une connexion réseau. Pour ce faire, vous allez utiliser un environnement virtuel émulant un réseau de machines sous *Debian*[1], c'est à dire la distribution que vous utilisez actuellement. Le but précis ce TP est de mettre en œuvre un exemple de type PKI afin de sécuriser les accès à un site web. L'environnement virtuel que nous allons utiliser est *NEmu*[2].

2 Principes

2.1 Chiffrement

Un moyen, massivement utilisé dans les systèmes actuels, pour sécuriser des connexions qui transportent des données sensibles est le chiffrement. En d'autres termes, l'idée est de crypter les données qui transitent entre 2 entités et de faire en sorte que seules ces mêmes entités puissent déchiffrer le contenu. Il existe pour cela 3 modèles de cryptographie :

— symétrique ;
— asymétrique ;
— hybride.

2.1.1 Symétrique

Le chiffrement symétrique est le moyen le plus intuitif. Pour une connexion donnée, il existe une clé k qui permet à la fois de chiffrer et de déchiffrer le contenu. Cette clé est partagée par les entités de la connexion. Il existe plusieurs algorithmes de chiffrement symétrique : DES, AES, XOR, etc.

2.1.2 Asymétrique

Le chiffrement asymétrique est un moyen plus évolué. Pour une connexion donnée, il existe un couple de clés $\{k,p\}$ qui permet de chiffrer et de déchiffrer le contenu. k permet de chiffrer des messages qui ne pourront être déchiffrés que par p (et réciproquement). k se trouve donc sur une entité, et p sur l'autre. Il existe plusieurs algorithmes de chiffrement asymétrique : RSA, DSA, ECC, etc.

2.1.3 Hybride

Le chiffrement hybride fait appel à la fois à la cryptographie symétrique et asymétrique. Un expéditeur génère une clé symétrique aléatoire, il chiffre ensuite cette clé avec la clé publique du destinataire et l'envoie. Les messages suivants seront chiffrés grâce à cette clé symétrique. L'intérêt d'utiliser cette méthode réside dans le fait qu'un cryptage symétrique est plus rapide qu'un cryptage asymétrique.

2.2 Intégrité

Un message, même crypté, peut être modifié de manière aléatoire par un attaquant qui se positionnerait en homme du milieu. Par conséquent, une empreinte (ou résumé) est ajoutée au message chiffré. Une empreinte est calculée grâce à une fonction de hachage et possède une longueur faible ($\leq$ 512 *bits*). Il existe plusieurs fonctions de hachage : MD5, RIPEMD, SHA-1, SHA-256, SHA-512, etc.

2.3 Authentification

Afin de renforcer l'authenticité de la source d'un message chiffré, on peut lui ajouter une signature. Cette signature correspond en réalité à chiffrer l'empreinte avec une clé privée. Par conséquent, pour chiffrer une communication entre plusieurs entités, il faudra une paire de clés par entité et que chacune d'entre elles possède l'ensemble des clés publiques.

2.4 Modèles de sécurisations

Il existe principalement deux modèles de sécurisation sur Internet.

1. http://www.debian.org
2. http://nemu.valab.net

Vincent Autefage
autefage@labri.fr

2.4.1 Le modèle hiérarchique

Ce modèle, appelé aussi PKI (*Public Key Infrastructure*), est utilisé afin de garantir l'authenticité d'une source de manière centralisée. Une autorité, que l'on nomme CA, génère un couple de clés. Elle signe sa clé publique à l'aide de sa clé privée. Cette clé publique est transmise aux clients en hors-ligne. Le CA génère autant de paires de clés uniques qu'il y a d'entités. La clé publique de l'entité est signée avec la clé privée du CA. Une clé publique signée avec une clé privée est appelé un *certificat*.

Ce modèle est utilisé pour les connexions HTTPS, il est aussi appelé TLS (anciennement SSL). La certification est payante dans la mesure où les CA sont des entreprises de services.

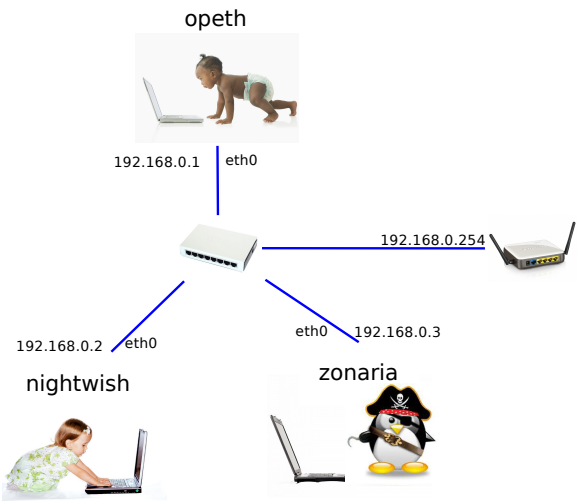
2.4.2 Le modèle de confiance

Ce modèle, aussi appelé PGP (*Pretty Good Privacy*), est également utilisé pour garantir l'authenticité d'une source mais de manière distribuée. Il n'y a pas d'autorité de certification, les certificats sont donc auto-signés. L'authentification est donc plus ou moins relative.

Ce modèle est principalement utilisé dans les échanges de mails sécurisés.

3 Le réseau

Nous allons travailler sur le réseau suivant :



Nous pouvons constater que ce réseau est composé de 3 machines et un routeur DHCP/DNS. Les 3 machines nommées sont des terminaux utilisateurs standards tournant sous *Debian* et inter-connectées grâce à un switch sur lequel une *box* est présente. Les machines virtuelles vous sont livrées *nues*. C'est à dire qu'elles disposent uniquement des réglages élémentaires du système. Le mot de passe *root* est **plop**.

4 Avant de commencer...

— Pour lancer le réseau virtuel :
```
$ source /net/ens/vince/virt/nemu-init.rc
$ nemu-kvm start
$ nemu-vnet /net/ens/vince/virt/config/netcrypt.py
```
— Pour quitter le réseau virtuel, tapez **quit()** dans le terminal principal.
— Pour sauvegarder le réseau virtuel, tapez **save()** et validez dans le terminal principal. Le réseau sera sauvegardé dans ∼ */netcrypt.tgz*.
— Pour redémarrer (violemment) le réseau virtuel, tapez **reboot** et validez dans le terminal principal.
— Pour restaurer le réseau virtuel précédemment sauvegardé :
```
$ nemu-restore ~/netcrypt.tgz
```
— Les éditeurs *jed*[3], *nano*[4] et *vi*[5] sont installés sur le système.

3. http://www.jedsoft.org/jed
4. http://www.nano-editor.org
5. http://vim.sourceforge.net

4.1 Amorçage du réseau

1) Lancez le réseau virtuel comme indiqué ci-dessus. Trois fenêtres correspondant aux consoles de chacune des machines devraient apparaître.

4.2 Configuration générale

2) Configurez les interfaces réseaux de *nightwish*, *zonaria* et *opeth* à l'aide de la commande **dhclient**.

Rappels :

```
# dhclient <iface>
```

3) Vérifiez que la configuration est effective à l'aide des commandes **ifconfig**, **route** et **ping**.

Rappels :

```
# ifconfig <iface>
# route -n
# ping <@IP>
```

5 Site web sécurisé

Les opérations suivantes seront effectuées sur *opeth*. Elles requiert une suite logicielle nommée **openssl**[6].

5.1 Génération des certificats

4) Créez un répertoire *ssl* dans le répertoire de configuration du serveur web **lighttpd**, et rendez vous y.

```
# mkdir /etc/lighttpd/ssl
# cd /etc/lighttpd/ssl
```

5) Générez le couple de clés pour votre site web (*common name* est le nom DNS du site web).

```
# openssl genrsa -out site.key 1024
# openssl req -new -key site.key -out site.csr
```

6) Vous allez vous même vous placer en CA (sur *opeth*) et donc générer un couple de clés primaires (les informations saisies doivent être différentes de celles du site web).

```
# openssl genrsa -out ca.key 2048
# openssl req -new -x509 -days 3650 -key ca.key -out ca.crt
```

7) Signez votre clé publique avec la clé privée du CA.

```
# openssl x509 -req -in site.csr -out site.crt -sha1 -CA ca.crt
               -CAkey ca.key -CAcreateserial -days 3650
```

8) Le contenu d'un certificat est peu lisible. Mais vous pouvez faire appel à **openssl** pour le rendre plus *attrayant*...

```
# cat <cert>
# openssl x509 -in <cert> -text
```

9) Créez un fichier *pem* pour le serveur web. Ce type de fichier contient à la fois le certificat du serveur ainsi que sa clé privée.

```
# cat site.key site.crt > site.pem
```

6. http://openssl.org

5.2 Configuration du site web

10) Écrivez une page simple dans le répertoire **/var/www** de *opeth*.

11) Activez SSL/TLS sur le serveur web. Pour cela, ajoutez les lignes suivantes dans */etc/lighttpd/lighttpd.conf*.

```
$SERVER["socket"] == "0.0.0.0:443" {
     ssl.engine   = "enable"
     ssl.pemfile  = "/etc/lighttpd/ssl/site.pem"
     ssl.ca-file  = "/etc/lighttpd/ssl/ca.crt"
}
```

12) Lancez le serveur web à l'aide de la commande suivante.

```
# service lighttpd start
```

13) Essayez d'accéder au site web de *opeth* depuis *nightwish* en HTTPS.

14) Quel est le problème ?

15) Corrigez le en ajoutant une exception à **Firefox**. Vous pouvez consulter la liste des certificats de CA valides dans *Edit/Preferences/Advanced/Certificates/View Certificates/Servers*.

16) Ré-essayez d'accéder au site web de *opeth* depuis *nightwish* en HTTPS.

5.3 Tentative d'espionnage

17) Mettre en place une procédure de *man in the middle* de *zonaria* sur *nightwish* avec **arpspoof** et **wireshark**.

18) Ré-essayez d'accéder au site web de *opeth* depuis *nightwish* en HTTPS.

19) Essayez d'espionner le contenu de la communication entre *nighwtsih* et *opeth* depuis *zonaria* ?

20) Cela fonctionne-t-il ?

21) Stoppez la procédure d'espionnage sur *zonaria*.

22) Éteignez chaque machine correctement à l'aide de la commande **halt**.

23) Clôturez l'environnement virtuel à l'aide de la commande **quit()**.

Vincent Autefage
autefage@labri.fr