UNIVERSITÉ de Poitiers

Institut des Risques Industriels, Assurantiels et Financiers

TP RESEAU N°3

Mise en place du système d'un serveur Apache-SSL

Master Management de la Sécurité des Systèmes Industriels et des Systèmes d'Information
année 2005/2006

Julien VEHENT

1. Introduction

Ce compte rendu décris la mise en place d'un accès HTTPS avec Apache 2. Nous n'aborderons pas la configuration d'Apache pour autre chose que HTTPS. L'objectif est qu'à la fin de ce TP, le serveur soit capable d'accepter une connexion HTTPS sur le port 443, de négocier une session TLS et d'afficher un contenu quelconque.

Secure Socket Layer (SSL), maintenant remplacé par Transport Layer Security (TLS), est un protocole de niveau légèrement supérieur à la couche transport où officie TCP. En fait, TLS va intercepter les données HTTP, y appliquer ses algorithmes de chiffrement et passer cela à TCP assurant ainsi la sécurité des données. TLS inclut également le mécanisme de négociation de connexion appelé *Handshake TLS*. Les détails de ce protocole ne sont pas l'objet du présent TP.

N'ayant pas de Fedora core 4 chez moi, j'ai reproduit le travail mené en cours sur un système Debian. Les différences sont mineures sauf pour le fichier ssleay.cnf inexistant sous Fedora et présent lors de l'installation sous Debian.

2. Apache2 et mod_ssl

Apache2 utilise mod_ssl pour HTTPS. Le module se base sur OpenSSL pour les outils cryptographiques. Il faut l'activer dans la config d'Apache2 :

```
root@locahost:/etc/apache2/mods-available# ls -al |grep ssl

-rw-r--r--  1 root root 3545 2006-03-27 13:16 ssl.conf
-rw-r--r--  1 root root   58 2006-03-27 13:16 ssl.load
```

Dans le répertoire /etc/apache2, le dossier mods-available contient ssl.load avec la ligne qui permet de lancer le module SSL:

```
# cat ssl.load
LoadModule ssl_module /usr/lib/apache2/modules/mod_ssl.so
```

Le fichier ssl.conf, présent dans le même dossier, contient la configuration du module ssl, que nous verrons juste après.
En premier lieu, dans le fichier apache2.conf on spécifie que tout le répertoire mods-enable est chargé au lancement du service :

```
# cat /etc/apache2/apache2.conf |grep mods-enable
Include /etc/apache2/mods-enabled/*.load
Include /etc/apache2/mods-enabled/*.conf
```

Cette configuration est faite par défaut sous Debian. Pour lancer le module SSL au lancement du logiciel Apache2, il suffit de faire un lien symbolique des deux fichiers qui sont dans mods-available vers mods-enable.

 # ls -al |grep ssl
[...]root 26 2005-10-16 01:34 ssl.conf -> ../mods-available/ssl.conf
[...]root 26 2005-10-16 01:34 ssl.load -> ../mods-available/ssl.load

Maintenant que le lancement du module est géré, nous pouvons nous intéresser à sa configuration. Pour cela, nous commençons par en faire un backup et un petit nettoyage:

root@localhost:mods-available# mv ssl.conf ssl.conf.old
root@localhost:mods-available# awk '! (/^ *#/ || /^$/) { print $0 }' ssl.conf.old > ssl.conf

« awk » est un puissant processeur de texte qui ne sera pas détaillé ici.
Entrons dans le fichier pour l'éditer. La ligne vraiment importante est ***SSLCipherSuite*** car elle définit les différents algorithmes utilisés.
Nous souhaitons assurer un bon niveau de sécurité en bannissant les algorithmes liés à la deuxième version du protocole SSL et connus pour leurs failles. En fait, nous voulons nous limiter à SSLv3 et TLSv1.
Nous modifions donc la ligne du fichier ssl.conf comme suis:

SSLCipherSuite HIGH:MEDIUM:!aNULL:+SHA1:+MD5:+HIGH:+MEDIUM

Voici les algorithmes qui, grâce a la ligne ci-dessus, pourront être utilisés:

 # openssl ciphers -v 'HIGH:MEDIUM:!aNULL:+SHA1:+MD5:+HIGH:+MEDIUM'

DHE-RSA-AES256-SHA	SSLv3	Kx=DH	Au=RSA	Enc=AES(256) Mac=SHA1
DHE-DSS-AES256-SHA	SSLv3	Kx=DH	Au=DSS	Enc=AES(256) Mac=SHA1
AES256-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=AES(256) Mac=SHA1
EDH-RSA-DES-CBC3-SHA	SSLv3	Kx=DH	Au=RSA	Enc=3DES(168) Mac=SHA1
EDH-DSS-DES-CBC3-SHA	SSLv3	Kx=DH	Au=DSS	Enc=3DES(168) Mac=SHA1
DES-CBC3-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=3DES(168) Mac=SHA1
DES-CBC3-MD5	SSLv2	Kx=RSA	Au=RSA	Enc=3DES(168) Mac=MD5
DHE-RSA-AES128-SHA	SSLv3	Kx=DH	Au=RSA	Enc=AES(128) Mac=SHA1
DHE-DSS-AES128-SHA	SSLv3	Kx=DH	Au=DSS	Enc=AES(128) Mac=SHA1
AES128-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=AES(128) Mac=SHA1
DHE-DSS-RC4-SHA	SSLv3	Kx=DH	Au=DSS	Enc=RC4(128) Mac=SHA1
RC4-SHA	SSLv3	Kx=RSA	Au=RSA	Enc=RC4(128) Mac=SHA1
RC4-MD5	SSLv3	Kx=RSA	Au=RSA	Enc=RC4(128) Mac=MD5
RC2-CBC-MD5	SSLv2	Kx=RSA	Au=RSA	Enc=RC2(128) Mac=MD5
RC4-MD5	SSLv2	Kx=RSA	Au=RSA	Enc=RC4(128) Mac=MD5

Les colonnes sont:

- le nom de la suite de chiffrement
- le numéro de version (imprécis ici, en fait il s'agit de TLSv1)
- l'algorithme d'échange de clé (Kx=)
- l'algorithme asymétrique d'authentification (Au=)
- l'algorithme de chiffrement symétrique (Enc=)
- l'algorithme d'empreinte (Mac=)

Ajoutons également à ce fichier le chemin de notre certificat auto-signé que nous allons générer

dans la partie suivante:

SSLCertificateFile /etc/apache2/ssl/mssisi-web.pem

On peut désormais fermer le fichier ssl.conf.

Maintenant, il nous faut créer un « *virtual host* » qui va appeler notre site par défaut mais avec une connection ssl. Recopions les paramétres du virtual host classique:

 # cat sites-available/default > sites-available/default-ssl

Il faut modifier le nom du virtual host, ici mssisi.org:443, et ajouter les lignes suivante juste après la déclaration du vhost:

SSLEngine on
ErrorLog /var/log/apache2/error-ssl.log
CustomLog /var/log/apache2/access-ssl.log combined

et on link le virtual host ssl vers le répertoire sites-enabled:

#ln -s /etc/apache2/sites-available/default-ssl /etc/apache2/sites-enabled/default-ssl

Un dernier point pour la config d'apache2 est le fichier ports.conf. Il est obligatoire de lui indiquer d'écouter sur le port 443, le port TLS.

 # echo 'Listen 443'>>/etc/apache2/ports.conf
 # cat /etc/apache2/ports.conf
Listen 80
Listen 443

3. Certificat auto-signé

Pour utiliser TLS, il nous faut un certificat au format X509 et une clé privée. Les outils OpenSSL nous permettent de générer un certificat auto-signé qui suffira pour nos besoins. Pour cela, il nous faut en premier lieu un fichier allégé de configuration de SSL: le fichier ssleay.cnf non disponible par défaut sous Fedora mais qui l'est sous Debian. Voici la version commentée du fichier:

 # cat /usr/share/apache2/ssleay.cnf

```
#
#  SSLeay exemple de fichier de configuration
#
RANDFILE              = $ENV::HOME/.rnd
# les champs du certificat que l'on demandera a sa création
[ req ]
default_bits                  = 1024
default_keyfile               = privkey.pem
distinguished_name            = req_distinguished_name
```

```
# les champs suivants sont généralement du texte, ils définissent la carte d'identité de
#son propriétaire
[ req_distinguished_name ]
countryName                     = Nom du Pays (obligatoirement 2 lettres)
countryName_default             = FR
countryName_min                 = 2
countryName_max                 = 2
# on demande le département
stateOrProvinceName             = Departement
# et on fournit une valeur par défaut au cas ou l'utilisateur ne voudrais pas s'embeter
stateOrProvinceName_default             = Deux-Sevres
# pareil pour la suite....
localityName                    = Nom de la ville
organizationName                = Nom de l'organisation
organizationName_max            = 64
organizationalUnitName          = Nom de la section de l'organisation
organizationalUnitName_max              = 64
#
# Common name est un champ très important quand on génère un certificat pour un site
#web.  En effet, si ce champ ne correspond pas a l'url du site web, le navigateur formule
#une erreur d'usurpation d'identité
commonName                      = nom du serveur /!\ METTRE L'URL
commonName_max                    = 64
#l'adresse email, au cas ou.....
emailAddress                  = Email
emailAddress_max                = 40
```

Maintenant que l'on a ce fichier, on peut générer notre certicat auto-signé. On travaille dans le répertoire */etc/apache2/ssl/ :*

```
#openssl req -config /usr/share/apache2/ssleay.cnf -new -x509 -nodes \
-out mssisi-web.pem -keyout mssisi-web.pem -days 1024

Generating a 1024 bit RSA private key
....++++++
.....................................................................++++++
writing new private key to '/etc/apache2/ssl/mssisi-web.pem'
-----

Nom du Pays (obligatoirement 2 lettres) [FR]:FR
Departement [Deux-Sevres]:2sevres
Nom de la ville []:Niort
Nom de l'organisation []:mssisi
Nom de la section de l'organisation []:security department
nom du serveur /!\ METTRE L'URL []:www.mssisi.org
Email []:julien@mssisi.org
```

Cette commande crée deux fichiers que sont la clé privée et le certificat public auto-signé.

Les arguments passés à l'exécutable openssl sont:

- req : désigne une requête de certificat
- -config : le fichier de configuration utilisé pour générer le certificat
- -new : désigne une requête sans rapport de précédence
- -x509 : désigne la structure de champs du certificat
- -nodes : demande de ne pas chiffrer le clé privée, sinon Apache2 ne pourra pas la lire
- -out : chemin du certificat autosigné
- -keyout : chemin de la clé privée
- -days : durée de validité des clés

On commence par changer les droits sur ces deux fichiers:

#chmod 600 mssisi-web.*

Maintenant que la configuration est terminée, il faut relancez apache2 et s'assurer qu'il démarre sans aucun autre message que:

 # /etc/init.d/apache2 start
Starting web server: Apache2.

On peut maintenant se connecter au site web https://www.mssisi.org, la sécurité de la liaison est complètement assurée par le protocole TLS.

Comme le certificat que nous utilisons est auto-signé, les navigateurs internet affichent un avertissement de sécurité lors de la connexion. Le moyen de résoudre ce problème est d'utiliser un certificat émis par une PKI.