Sécurisation des applications avec SSL

SSL/TLS

Parmi les outils susceptibles de garantir le cryptage, *SSL (Secure Socket Layer : Couche de socket sécurisé)*, défini par la société Netscape en 1994 (V.1) et déployé publiquement en version 2 (1995) s'est imposé comme la technique de sécurisation des échanges en environnement internet.

Il s'applique à tous les protocoles de la couche *application* (couche 7) puisqu'il se positionne plus bas dans le modèle *OSI* (couche 5 : Session).

Il supporte les *certificats X.509*.

TLS (Transport Layer Security) est le nom du protocole depuis 2001, qui remplace la version 3.1 de SSL. Il est maintenant défini par l'*IETF (Internet Engineering Task Force)* qui se charge de normaliser les techniques utilisées sur Internet.

La mise en place d'un service sécurisé par SSL/TLS permettra d'éviter que l'on puisse exploiter le contenu en cas d'interception des échanges entre le client et le serveur.

Procédure

La démarche est commune quel que soit le service que l'on souhaite sécuriser. Seuls les fichiers à modifier changent d'un service à l'autre.

Créer le répertoire pour stocker les clés et certificats

1. Créer la clé privée
2. Créer le certificat X509 à partir de la clé privée en renseignant toutes les informations demandées
3. Paramétrer le service concerné (apache, proftpd, etc)
    - activer le module,
    - configurer les fichiers pour qu'ils utilisent la clé et le certificat
    - mettre en écoute sur les ports spécifiques
    - redémarrer le service pour prendre en compte les modifications
4. Tester depuis un client en lui indiquant l'adresse et le port adéquats
    - Éventuellement, accepter le certificat auto-signé (attention, cela peut être un danger si on ne connaît pas la source).

Mode opératoire

OpenSSL est une transposition *Open Source* (http://www.openssl.org/) des **préconisations de l'IETF** pour la mise en place d'une couche sécurisée.

Le nom est resté *OpenSSL* mais la bibliothèque supporte aussi la version TLS.

OpenSSL gère de nombreux algorithmes de cryptage (AES, DES, RSA,...) et algorithmes de hachage (SHA, MD5, ...).

Installation des outils

Installer la bibliothèque OpenSSL

```
apt install openssl
```

Installer les outils de manipulation des éléments de sécurité

```
apt install easy-rsa
```

Cet outil propose notamment des commandes pour la création de clé secrète (*gendh*, *genrsa*, ...) ou la génération de certificat (*x509*).

Mise en place d'un certificat avec OpenSSL

1 : Création d'une clé privée

La base des techniques de chiffrement asymétrique repose sur la présence, à un endroit unique, d'une ***clé privée***. Elle permettra de créer une ***clé publique*** (dans un ***certificat***) et de décrypter ce qui aura été chiffré par cette ***clé publique***.

On utilisera (pour l'algorithme RSA) la commande ***genrsa*** de *OpenSSL*.

```
openssl genrsa -out <nom_fichier_cle> <taille_cle>
```

Exemple : créer une clé privée dans le dossier private

```
cd /etc/ssl/private
openssl genrsa -out cleGSB.key 2048
```

On pourra alors éditer la clé (avec nano) et en lire le contenu (évidemment inaccessible). Cette même clé peut servir à établir des certificats différents pour des usages distincts (serveur FTP, serveur Web, messagerie, etc).

2 : Création d'un certificat X509

La clé privée ne doit jamais être diffusée

On va donc générer la ***partie publique*** sous forme d'un certificat.

Pour que ce dernier soit accessible aux navigateurs, on a recours au format standard X509.

Les certificats sont à stocker dans le dossier ***certs*** du dossier SSL.

La création va donc passer par deux étapes :

- Création d'un d'une demande de signature de certificat (Certificate signing request) : on devra renseigner les coordonnées de l'entreprise

  si la signature est faite par la même machine que la demande, le certificat est auto-signé et génèrera une alerte sur les navigateurs

```
openssl req -new -key <nom_fichier_cle> -out <nom_certif_generic.csr>
```

- signature et création du certificat au X509

```
openssl x509 -req -days <nb_jours> -in <nom_certif_generic.csr> -signkey
<nom_fichier_cle> -out <nom_certif_X509.crt>
```

Exemple : création d'un certificat dans le dossier des certificats SSL à partir de la clé RSA

```
cd /etc/ssl/certs
openssl req -new -key ../private/cleGSB.key -out GSBCertGen.csr
openssl x509 -req -days 365 -in GSBCertGen.csr -signkey
../private/cleGSB.key -out GSBcertif.crt
```

Penser à adapter les valeurs à votre environnement

Le certificat est alors prêt.

Il reste à configurer les services susceptibles de s'appuyer sur ce certificat.

Configuration d'Apache avec SSL/TLS

Activation du module SSL

La configuration nécessite l'activation du module SSL pour Apache2

```
a2enmod ssl
```

remarque : Si le module est déjà activé, un message l'indique.

Si la commande a2enmod ne fonctionne pas, faire les commandes suivantes à la place :

- cp /etc/apache2/mods-available/ssl.load /etc/apache2/mods-enabled/
- cp /etc/apache2/mods-available/ssl.conf /etc/apache2/mods-enabled/
- cp /etc/apache2/mods-available/socache_shmcb.load /etc/apache2/mods-enabled/
- systemctl restart apache2 **ou** service apache2 restart

Contrôle

On peut alors voir dans le fichier ** */etc/apache2/mods-enabled/ssl.load* ** que la ligne ci-dessous est dé-commentée

```
LoadModule ssl_module /usr/lib/apache2/modules/mod_ssl.so
```

Dans le fichier *ports.conf* de Apache, on vérifiera que le serveur écoute sur le port standard 443 (ou sur un autre port si on souhaite faire une configuration personnalisée).

Prise en charge des éléments de sécurité

Se placer dans le dossier // /etc/apache2/sites-enabled// et ensuite éditer le fichier *000-default.conf* (ou dans // /etc/apache2/httpd.conf//).

```
cd /etc/apache2/sites-enabled
nano 000-default.conf
```

On ajoutera un hôte virtuel pour cette écoute :

```
#on adaptera le numéro de port conformément à ce qui a été écrit dans
ports.conf
<VirtualHost *:443>
    DocumentRoot /var/www/html
        # active le SSL
    SSLEngine on
        # chemin du certificat X509
    SSLCertificateFile /etc/ssl/certs/GSBCertif.crt
        # chemin de la clé privée
    SSLCertificateKeyFile /etc/ssl/private/cleGSB.key
</VirtualHost>
```

Il faudra redémarrer Apache, qui indiquera si une erreur éventuelle est rencontrée (dans le chemin, dans le nom du fichier, dans le contenu du certificat, etc).

```
systemctl restart apache2
```

Contrôle depuis un navigateur

Dans la barre de navigation du navigateur, on tapera https://<adresseServeur>.

Du fait que le certificat que nous avons créé n'est pas garanti par un organisme officiel (on parle d'autorité), le navigateur met en garde sur le manque de confiance accordée à un certificat signé par son créateur (auto-signé / self-signed).

On doit vérifier qu'un cadenas fermé au bas du navigateur atteste que la navigation est malgré tout sécurisée par du chiffrement.

Redirection de HTTP vers HTTPS

Lorsqu'on met en place une sécurisation SSL d'un serveur WEB, il peut être intéressant de laisser possible l'accès HTTP (pour les utilisateurs étourdis), mais en forçant le renvoi vers la version HTTPS.

Pour cela, dans le ***VirtualHost*** du port 80, on ajoutera la ligne suivante :

```
<VirtualHost *:80>
//ligne à ajouter en adaptant l adresse du serveur
Redirect permanent / https://<IP_ou_FQDN_SERVEUR>/

</VirtualHost>
```

Configuration de ProFTP avec SSL/TLS

Activation du module TLS

Pour commencer, on devra indiquer dans le fichier */etc/proftpd/modules.conf* qu'il faut activer TLS

```
LoadModule mod_tls.c
```

On ira ensuite préciser au fichier *proftpd.conf* d'inclure le fichier de configuration de TLS

```
nano /etc/proftpd/proftpd.conf
```

On dé-commentera la ligne suivante :

```
Include /etc/proftpd/tls.conf
```

Prise en charge des éléments de sécurité

La configuration se passe dans le fichier tls.conf:

```
nano /etc/proftpd/tls.conf
```

Dans ce fichier *tls.conf* on devra trouver au minimum les éléments :

```
<IfModule mod_tls.c>
    #active le TLS
        TLSEngine on
        # dossier pour enregistrer les journaux tls
    TLSLog /var/log/proftpd/tls.log
        # versions supportées (2 et 3)
    TLSProtocol SSLv23
        #chemin du certificat
    TLSRSACertificateFile /etc/ssl/certs/GSBCertif.crt
        # chemin de la clé
    TLSRSACertificateKeyFile /etc/ssl/private/cleGSB.key
        # n'oblige pas l'authentification des clients pour TLS
    TLSVerifyClient off
        # peut obliger les clients à utiliser TLS
    #TLSRequired on
</IfModule>
```

Il faudra bien entendu redémarrer le service *proftpd* qui indiquera si une erreur est rencontrée (dans le chemin, dans le nom du fichier, dans le contenu du certificat, etc).

```
systemctl restart proftpd
```

Test depuis le client

On accèdera depuis un client FTP en contactant le serveur par une connexion *FTP SSL/TLS Explicite* (ici sous FileZilla Client).

On rencontrera une mise en garde indiquant que le certificat n'étant pas garanti par un tiers (il est auto-signé).

Sources

- http://httpd.apache.org/docs/trunk/fr/ssl/ssl_intro.html : sur le site de Apache, une explication détaillée des principes du chiffrement, du rôle d'un certificat, des déclinaisons de SSL/TLS

From:
https://wiki.sio.bts/ - **WIKI SIO : DEPUIS 2017**

Permanent link:
https://wiki.sio.bts/doku.php?id=ssl&rev=1795432896

Last update: **2026/11/23 11:21**