Configuration d'Apache avec SSL

SSL : Secure Socket Layer est un protocole de sécurisation des couches applicatives. Il fonctionne avec de nombreux protocoles, mais son usage le plus répandu est la sécurisation de http (https). SSL apporte des services d'authentification, de confidentialité et de contrôle d'intégrité.

1. Cryptographie et certificats

Une explication exhaustive de la cryptographie et des certificats numériques X509 dépasserait de beaucoup les objectifs de ce livre. Toutefois, l'utilisation de certificats est obligatoire pour sécuriser l'accès aux pages web d'un serveur par SSL (https).

a. Concepts cryptographiques

Toute infrastructure cryptographique repose sur des algorithmes de cryptage. Ces algorithmes appartiennent forcément à trois familles distinctes : les algorithmes symétriques où on crypte et décrypte avec une clé unique, les algorithmes asymétriques, où l'on dispose d'une paire de clés, l'une pour crypter et l'autre pour décrypter, et enfin les algorithmes de hachage, à sens unique et n'exploitant pas de clé de cryptage.

La cryptographie asymétrique exploite deux clés. Par convention, on décide qu'une de ces clés sera privée et ne devra être exploitée que par son propriétaire, et l'autre sera publique, et pourra être vue de tous, même des personnes hostiles. La consommation importante en ressources processeur de la cryptographie asymétrique la rend inapte au cryptage de grandes quantités de données, mais les nombreuses possibilités offertes par les clés publiques et privées en font un outil incontournable. La clé publique servira au cryptage de petites données (comme d'autres clés, symétriques par exemple), alors que la clé privée sera utilisée pour les opérations de signature numérique (on signe avec quelque chose qui n'appartient qu'à soi).

b. Les certificats numériques X509

Si les algorithmes utilisés couramment sur Internet et dans les entreprises sont réputés fiables, l'analyse des systèmes cryptographiques montre que la vulnérabilité vient souvent des risques liés à la transmission de la clé publique d'un utilisateur ou d'un serveur. La conception même de la cryptographie asymétrique fait que cette clé est strictement inutilisable seule, et qu'elle ne permet en rien de déduire la valeur de la clé privée. Toutefois, les failles des échanges confidentiels ou des signatures numériques de documents reposent toutes sur des clés publiques dont le nom du propriétaire serait usurpé. En clair, une clé publique circule, mais ce n'est pas la bonne, et un intrigant fait passer sa clé publique pour celle d'un autre. La personne trompée pourrait alors crypter des éléments très confidentiels avec la mauvaise clé et donc mettre ces informations en danger.

Les certificats numériques X509 ont pour but d'établir de façon formelle un lien entre une identité (nom, adresse IP, etc.) et une clé publique. Les certificats ne peuvent être contrefaits car ils sont signés par un tiers en qui toutes les parties placent leur confiance. Ces tiers sont appelés « Certificate Authority » (autorité de certification). Ils peuvent être publics et reconnus de tous ou privés et utilisés dans un cadre restreint. Dans ce cas-là, les applications clientes devront être configurées pour reconnaître l'autorité de certification qui aura émis les certificats.

Dans le cadre de l'utilisation de certificat pour un serveur web, le serveur doit disposer d'un certificat qui atteste de son identité : une clé publique liée à son nom d'hôte. Lors d'une connexion https de la part d'un navigateur, le serveur envoie son certificat, le navigateur en vérifie la validité, puisque le nom sous lequel on accède au serveur est bien celui annoncé dans le certificat. Dans le cas contraire, le navigateur oppose une alerte de sécurité. L'utilisateur est alors libre de laisser la connexion sécurisée s'établir ou non, mais sans savoir si le serveur auquel il s'adresse est légitime ou s'il s'agit d'un usurpateur.

c. Génération locale d'un certificat

Le fonctionnement de HTTP avec SSL requiert qu'un certificat contenant la clé publique du serveur web soit envoyé au navigateur client et que cette clé publique soit toujours envoyée sous forme de certificat. Apache configuré pour SSL doit donc disposer d'un certificat qu'il pourra envoyer à ses clients.

Le certificat utilisé pour Apache pourra être fourni par une autorité de certification publique ou privée fournie par une application spécialisée. Dans cette hypothèse, l'autorité fournira un certificat qui sera exporté sous forme de fichier, copié sur le disque du serveur Apache, et exploité par le serveur.

Dans le cas d'un usage ponctuel ou à des fins de test, on peut aussi générer localement un certificat prêt à l'emploi qui sera exploitable par le serveur Apache. Il existe des utilitaires spécialisés dans cette fonction, mais qui sont généralement liés à une distribution, ou on peut le créer de toutes pièces avec les utilitaires de la bibliothèque openssl. Pour des besoins limités au test de la configuration ssl d'Apache, on choisira la solution la plus simple, à savoir utiliser les mêmes clés pour générer le certificat et pour le signer.

Génération du certificat auto-signé

```
openssl req -x509 -nodes -newkey rsa:taille -keyout fichier_clé -out fichier_certificat
```

Commande openssl pour génération de certificat : options et paramètres	
req	Demande de certificat.
-x509	On souhaite un certificat auto-signé et non une demande de signature.
-nodes	La clé de serveur ne doit pas être protégée par un mot de passe.
-newkey rsa:*taille*	On crée de nouvelles clés asymétriques RSA dont la taille est donnée en nombre de bits.
-keyout *fichier_clé*	Le fichier qui contiendra la clé privée du serveur.
-out *fichier_certificat*	Le fichier qui contiendra le certificat du serveur.

La commande ci-dessus génère la demande de certificat. Les champs normalisés du certificat sont demandés interactivement à l'utilisateur. La plupart de ces champs sont informatifs, mais dans le cadre de l'utilisation de certificat pour un serveur web, le champ Common Name (nom commun) doit impérativement correspondre au nom DNS qui figurera dans l'URL d'accès au serveur. Dans le cas contraire, le navigateur du client opposera une alerte de sécurité lors de la vérification du certificat de serveur.

Exemple de génération de certificat

Il est impératif de renseigner correctement le champ Common Name (CN). C'est celui qui sera associé formellement à la clé publique dans le certificat numérique.

```
root@serveur# openssl req -x509 -nodes -newkey rsa:1024 -keyout
serveur.cle -out certificat.pem
Generating a 1024 bit RSA private key
......++++++
.................++++++
writing new private key to 'serveur.cle'
-----
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]:FR
State or Province Name (full name) [Some-State]:France
Locality Name (eg, city) []:Lyon
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:R&D
Common Name (eg, YOUR name) []:192.168.1.1
Email Address []:
root@serveur#
```

2. Configuration ssl

a. Chargement du module SSL

Le module nécessaire au fonctionnement SSL est mod_ssl.

Chargement du module

```
LoadModule ssl_module /chemin/mod_ssl.so
```

Où *chemin* est le chemin absolu du fichier de module. Le chemin par défaut dépend de la façon dont Apache a été compilé et donc de la distribution.

b. Configuration des clés de serveur

Il faut ensuite préciser au serveur quelles sont les clés à utiliser pour son fonctionnement SSL. Il doit disposer de sa clé privée, et de son certificat qui contiendra sa clé publique.

Configuration des clés

```
SSLCertificateFile /chemin/fichier_certificat
SSLCertificateKeyFile /chemin/fichier_clé
```

Fichier apache2.conf : Déclaration des clés de serveur	
SSLCertificateFile	Désignation du fichier contenant le certificat de serveur.
SSLCertificateKeyFile	Désignation du fichier contenant la clé privée de serveur.

c. Gestion du fonctionnement SSL

Il n'y a plus qu'à demander au serveur d'écouter sur le port https, et à démarrer le moteur SSL qui une fois l'authentification réalisée, permettra le cryptage des échanges entre le client et le serveur.

Ouverture du port https

```
Listen 443
```

Activation du moteur SSL

```
SSLEngine on
```

Ces deux paramètres étant bien entendu à ajouter dans le fichier de configuration Apache.

d. Authentification des clients par certificat

Le fonctionnement SSL standard que nous venons d'établir est celui qu'on trouve sur Internet en général et satisfera à la plupart des besoins. Il est toutefois possible d'utiliser les certificats non pour transmettre une clé de session chiffrée et donc assurer la confidentialité mais pour garantir l'identité d'un client se connectant au serveur. Dans cette configuration, le navigateur client doit disposer d'un certificat qui sera vérifié par le serveur web. Pour ce faire, le serveur Apache doit posséder le certificat de l'autorité ayant émis les certificats clients.

Gestion de l'authentification par certificats dans le fichier de configuration Apache

```
SSLVerifyClient require
SSLCACertificateFile certificat-ca
```

Où *certificat-ca* représente le fichier de certificat d'autorité qui aura signé les certificats clients. Les certificats clients sont à installer dans le magasin de certificats du navigateur Internet.