Créer un certificat autosigné avec OpenSSL

Présentation

Un certificat autosigné peut servir :

- à un utilisateur pour lui permettre de s'authentifier auprès d'un serveur par exemple à distance avec SSH ;
- à un serveur pour configurer des communication chiffrées comme htpps pour un serveur Web.

Ce document utilise le logiciel OpenSSH soit sous Linux ou Windows.

- Pour en savoir plus sur l'utilisation d'OpenSSL en CLI :
  - https://wiki.openssl.org/index.php/Command_Line_Utilities
  - https://www.madboa.com/geek/openssl/

Démarche

La création d'un certificat auto-signé se réalise en trois étapes

- Création de la **clé privée** ;
- Création de la **requête de certification de la clé publique** ;
- Création du **certificat** qui va contenir la clé publique et les informations d'identification de l'utilisateur ou du serveur

Dans ce document est présenté la création d'un certificat autosigné pour un utilisateur

> Avec OpenSSL, la clé privée générée contient également les informations de la clé publique. Il n'est donc pas nécessaire de générer la clé publique séparément.

Création de la clé privée

Dans l'invite de commandes lancez la commande suivante :

```
$ openssl genrsa -aes256 -out user-private.pem 2048
```

Indiquez une pass-phrase pour votre nouvelle clé, pass-phrase qui sera demandé lors de la création du certificat.

Explication :

- **genrsa** : Génération de la clé avec l'algorithme RSA
- **-aes256** : Utilisation d'une passe phrase (facultatif) en précisant l'algorithme de chiffrement de la clé privée AES256
- **-out** : Indique le fichier de sortie
- **2048** : Taille de la clé

Pour visualiser le contenu de cette clé privée :

```
openssl rsa -in user-private.pem -text
```

> PEM (Privacy Enhanced Mail) est un format de fichier qui utilise l'encodage Base64 pour représenter des données binaires dans un format de texte ASCII. Ce format est couramment utilisé pour les certificats X.509 afin de faciliter leur transfert et leur stockage en tant que fichiers texte.

> La commande suivant permet de connaître la liste des chiffrements disponibles :
>
> ```
> $ openssl list -cipher-algorithms
> ```

Pour ne pas associer de pass-phrase à la clé privée ne précisez pas l'algorithme de chiffrement en lançant la commande :

```
$ openssl genrsa -out user-private.pem 2048
```

Générer la clé publique RSA (contenu dans le fichier user-private.pem)

```
$ openssl rsa -in user-private.pem -out user-public.pem -pubout
Enter pass phrase for user-private.pem
Writing RSA key
```

Pour visualiser le contenu de cette clé privée :

```
$ openssl rsa -in user-public.pem -pubin -text
```

Vous disposez maintenant :

- d'une clé privée **user-private.pem** ;
- d'une clé publique **user-public.pem**.

Il est possible de supprimer la pass-phrase associée à la clé privée en générant une nouvelle clé privée à partir de cela déjà créer. Pour cela exécutez la commande suivante :

```
$ openssl rsa -in user-private.pem -out user-private_nopass.pem
```

Création de la requête de demande de signature du certificat

Pour générer la requête de demande de signature, il faut utiliser le fichier contenant la clé privée.

Lors de la création de la requête vous avez à saisir vos informations d'identification et il est important d'indiquer un **Common Name** unique.

- Saisissez la commande suivante :

```
$ openssl req -new -key user-private.pem  -out user.csr

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]:Haute-Vienne
Locality Name (eg, city) []:Limoges
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Lycée Syzanne Valadon
Organizational Unit Name (eg, section) []:BTS SIO
Common Name (e.g. server FQDN or YOUR name) []:user
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Explication :

- **req** : Gestionnaire de requête
- **new** : Nouvelle requête
- **-x509** : indique le type de certificat Nouvelle requête
- **-key** : Indique la clé à utiliser
- **-out** : Indique le fichier de sortie

Génération du certificat auto-signé

Lors de la génération d'un certificat autosigné, vous avez à préciser le format du certificat (x509) et la durée de validation (ici 365 jours)

```
$ openssl x509 -req -days 365 -in user.csr -signkey user-private.pem -out user.crt

Signature ok
subject=C = FR, ST = Haute-Vienne, L = Limoges, O = Lyc\C3\83\C2\A9e Suzanne Valadon, OU = BTS SIO, CN =
user
Getting Private key
Enter pass phrase for user-private.pem:
sio@vpsdebian:~/certificat$
```

Le certificat généré est le fichier **.crt** et la clé privée associée, le fichier **.pem**.

Visualisation des informations du certificat

- en ligne de commande

```
$ openssl x509 -in user.crt -noout -text
```

- sous Linux, vous pouvez constater que le certificat est auto-signé :
- Sous Windows, après l'avoir récupéré avec scp, vous pouvez également constater que le certificat est auto-signé :

Utilisation de la clé publique pour une connexion distance en SSH

Génération de des informations qui seront ensuite copiée dans le fichier **~/.ssh/authorized_keys** du serveur distant:

```
# ssh-keygen -i -m PKCS8 -f user-public.pem >> id_rsa.pub
```

Le contenu du fichier **nom_fichier** doit être ajouter au contenu du fichier **~/.ssh/authorizedkeys**.

Vous pouvez“ ensuite vous connecter au serveur distant en indiquant votre clé privée :

```
$ ssh -i user-private.pem user@adresseIPserverur
```

Format de fichier

Lien : https://qastack.fr/server/9708/what-is-a-pem-file-and-how-does-it-differ-from-other-openssl-generated-key-file

- **.csr** : fichier de demande de signature d'un certificat contenant la clé publique et les information d'identité. Le format actuel est PKCS10, défini dans la RFC 2986. Après signature par l'autorité de certification et un certificat est renvoyé au demandeur.
- **.pem** (Privacy Enhanced Mail) : format de conteneur pouvant inclure :
  - uniquement le certificat public ,
  - une chaîne de certificats complète comprenant une clé publique, une clé privée et le certificat racine. De manière confuse ;
  - un CSR car le format PKCS10 peut être traduit en PEM. Le nom vient de Privacy Enhanced Mail (PEM).
- **.key** : fichier au format PEM contenant uniquement la clé privée d’un certificat spécifique. Il s’agit d’un nom conventionnel et non normalisé.
- **.pkcs12 .pfx .p12** : format de conteneur entièrement chiffré avec mot de passe contenant les paires de certificats publics et privés. Openssl peut transformer ce conteneur en un fichier .pem avec des clés publiques et privées (openssl pkcs12 -in file-to-convert.p12 -out converted-file.pem -nodes)
- **.der** : fichier dont les données sont encodées en binaire à la différence du format **.pem** codé en Base64. OpenSSL peut convertir un fichier .der en .pem (openssl x509 -inform der -in to-convert.der -out converted.pem). Windows considère les fichiers .der comme des fichiers de certificat et exportera par défaut les certificats sous forme de fichiers au format .der avec une extension différente.

- **.cert .cer .crt** : fichier au format .pem (ou rarement .der) portant une extension différente, reconnu par l'explorateur Windows comme un certificat, contrairement à **.pem**.
- **.p7b .keystore** - Défini dans le RFC 2315 en tant que PKCS numéro 7, il s'agit d'un format utilisé par Windows pour l'échange de certificats. Java les comprend de manière native et les utilise souvent .keystorecomme une extension. Contrairement aux certificats de style .pem, ce format comporte une méthode définie pour inclure les certificats de chemin de certification.
- **.crl** : liste de révocation de certificats utilisée par les autorités de certification comme moyen de désautoriser les certificats avant leur expiration.

En résumé, il existe quatre manières différentes de présenter les certificats et leurs composants:

- PEM : Régi par les RFC et utilisé préférentiellement par les logiciels open source. Il peut avoir une variété d'extensions (.pem, .key, .cer, .cert, etc.)
- **PKCS7** : Norme ouverte utilisée par Java et prise en charge par Windows. Ne contient pas de matériel de clé privée.
- **PKCS12** : offret une sécurité renforcée par rapport au format PEM en texte brut. Cela peut contenir du matériel de clé privée. Il est utilisé préférentiellement par les systèmes Windows et peut être librement converti au format PEM via OpenSL.
- **DER** : format parent de PEM et est à considéré comme une version binaire du fichier PEM codé en base64. Pas couramment utilisé en dehors de Windows.