Sécurité - TP
Certificats

Virginie Van den Schrieck

22 septembre 2015

L'objectif de ce TP est d'apprendre aux étudiants à gérer les certificats OpenSSL, c'est-à-dire à pouvoir les générer et les certifier.

Le temps prévu pour ce TP est d'environ trois heures, séance encadrée comprise, en binôme.

Un petit rapport documentant le travail effectué et répondant aux questions est demandé.

Ce TP est preque intégralement repris de : `http://www.fil.univ-lille1.fr/~wegrzyno/portail/PAC/Doc/TP-Certificats/index.html`

1 Préambule

1.1 Préparation

Etant donné que la matière liée au cours théorique n'a pas encore été donnée, il est fortement recommander de lire le chapitre 2 du syllabus avant de se lancer dans ce TP.

1.2 Contexte

Dans le cadre de ce TP, nous allons créer une paire de clés asymétriques. Ces clés servent en général pour l'autenthification. Nous nous baserons sur OpenSSL, qui est une boîte à outils cryptographiques implémentant les protocoles SSL et TLS offrant, d'une part, une bibliothèque de programmation en C, et d'autre part, un outil en ligne de commande qui permet de :

- Créer des clés RSA et DSA (signature)
- Créer des certificats X.509
- Calculer des valeurs de hachage (MD5, SHA, ...)
- Chiffrer ou déchiffrer (DES, IDEA, RC2, RC4, ...)
- Tester des configurations clients et serveurs SSL ou TLS
- Signer et chiffrer des courriers (S/MIME)

Cette paire de clé peut permettre par exemple d'authentifier un serveur Web. Nous allons donc, dans cette optique, certifier la clé publique du serveur

web en générant un certificat et en le faisant valider par une autorité de certification.

1.3 OpenSSL

OpenSSL est facilement disponible sur les systèmes Unix. Si vous ne disposez pas d'une machine personnelle Linux, démarrez une machine virtuelle (par ex. CentOS Sécurité), sur laquelle vous pourrez installer OpenSSL si ce n'est pas déjà le cas.

2 Création d'une paire clé publique/clé privée

2.1 Création de la clé

Nous allons tout d'abord générer une paire de clés RSA. Pour cela, tapez la commande suivante :

```
openssl genrsa -out <fichier> <taille>
```

`<fichier>` est le fichier dans lequel sera stocké la clé, et `<taille>` sera la longueur de la clé.

Par exemple :

```
openssl genrsa -out maCle.pem 1024
```

- *Expliquez ce qu'est le format PEM*
- *Est-ce que 1024 est une longueur de clé acceptable ? Justifiez*

2.2 visualisation de la clé

Visualisez le contenu de ce fichier à l'aide de la commande `cat`. Que voyez-vous ?

Il est possible d'obtenir plus d'informations sur les clés en les visualisant avec la commande `rsa` :

```
openssl rsa -in <fichier> -text -noout
```

- Expliquez chacune des options de la commande

2.3 chiffrement de la clé

Il n'est pas prudent de stocker une clé RSA en clair. Il est possible de la chiffrer, en utilisant un protocole de chiffrement symétrique comme DES, 3-DES ou idea. La clé secrète sera une *passphrase*.

Testez le chiffrement de la clé en utilisant la commande suivante :

```
openssl rsa -in maCle.pem -des3 -out maCleChiffree.pem
```

Essayez à présent d'afficher la clé, d'abord avec `cat`, puis avec `rsa`.

2.4 Extraction de la clé publique

Le fichier `maCle.pem` contient à la fois la clé privée et la clé publique. Il ne peut donc être utilisé tel quel, puisque la clé privée doit être conservée secrète. Pour extraire la clé publique, on effectue la commande suivante :

```
openssl rsa -in maCle.pem -pubout -out maClePublique.pem
```

Visualisez à présent le contenu du fichier ainsi créé, avec `cat` et `rsa`. Attention, dans ce cas, vous devez spécifier l'option `-pubin` pour indiquer qu'il s'agit d'une clé publique.

3 Chiffrement/Déchiffrement avec RSA

Nous allons à présent utiliser nos clés pour chiffrer des messages. La commande permettant le chiffrement est la suivante :

```
openssl rsautl -encrypt -in <msg> -inkey <key> -out <msg_chiffre>
```

Pour déchiffrer, il suffit de remplacer l'option `encrypt` par l'option `decrypt`. Si une clé publique est utilisée, il faut spécifier en plus l'option `-pubin`.

Vous allez à présent envoyer un fichier chiffré à votre binôme. Commencez par échanger vos clés publiques, puis créez un fichier texte contenant un message quelconque. Chiffrer le message, puis envoyez-le à votre binôme, qui fera de même. Essayez ensuite de déchiffrer le message que celui-ci vous a transmis.

4 Signature numérique

Nos clés asymétriques peuvent également servir pour signer des messages. Pour rappel, une signature numérique est une valeur de hachage d'un message, chiffrée avec la clé secrète de l'émetteur. Nous allons donc utiliser les fonctions de hachage disponibles avec OpenSSL. La commande est la suivante :

```
openssl dgst <hash_fn> -out <hash_value> <fichier_a_signer>
```

Les options pour les fonctions de hachage disponibles sont notamment `-md5` et `-sha1`.

Pour signer un document (ou ici, une valeur de hachage), on utilise la commande suivante :

```
openssl rsautl -sign -in <hash_value> -inkey <key> -out <signature>
```

Pour vérifier la signature, on utilisera :

```
openssl rsautl -verify -in <signature> -pubin -inkey <key> -out <hash_value>
```

Il reste alors à comparer la valeur de hachage calculée avec celle du fichier original.

Choisissez un fichier, puis signez-le. Observez la signature ainsi produite. Vérifiez-la.

5 Création d'un certificat

Nous allons à présent générer un certificat permettant de valider la clé publique d'un serveur Web.

Pour commencer, effectuez les opérations suivantes :

- Créez une paire de clés RSA de taille 2048, protégée par un mot de passe et stockée dans le fichier `maCle.pem`.
- Exportez la clé publique et stockez la dans le fichier `maClePub.pem`.

Vous allez à présent créer une requête pour un certicat, qui est en fait le certificat non encore signé par le CA.

Utilisez pour cela la commande suivante :

```
openssl req -new -key maCle.pem -out maRequete.pem
```

Observez le contenu de la requête avec `cat`, ou encore avec la commande :

```
openssl req -in monCertif.pem -text -noout
```

Quelles informations retrouvez-vous dans le certificat ?

6 Caractéristique du CA

Pour chiffrer notre certificat, il nous faut un CA. Le CA va signer le certificat sur base de sa propre paire de clés asymétriques. Dans le cadre de ce TP, nous utiliserons un CA fictif, appelée *Père Ubu*, dont vous trouverez sur le Campus Virtuel, d'une part, le certificat, et d'autre part, la paire de clés RSA.

Vous allez donc, grâce à ces deux fichiers, pouvoir jouer le rôle du CA et signer le certificat pour le serveur Web.

Examinez le certificat du CA, d'une part avec `cat`, d'autre part avec la commande suivante :

```
openssl x509 -in PereUbuCertif.pem -text -noout
```

- *Quelle est la date d'expiration du certificat du CA ?*
- *Et la longueur de sa clé ?*

7 Signature du certificat par le CA

Pour créer un certificat sur base d'une requête (ex : maRequete.pem), le CA effectue la commande suivante :

```
openssl x509 -days 10 -CAserial PereUbu.srl -CA PereUbuCertif.pem \
-CAkey PereUbuCle.pem -extfile usr.txt -extensions x509_ext \
-in maRequete.pem -req -out monCertif.pem
```

L'options `-days` spécifie la durée de la validité du certificat. PereUbu.srl contient le numéro de série du certificat à générer. usr.txt est un fichier texte contenant des informations sur l'utilisation prévue du certificat.

Vous allez à présent créer un certificat sur base de la requête créée pour le serveur web. Vous aurez besoin du mot de passe du CA pour utiliser la clé. Ce dernier est « merdre » [1].

Vérifiez le contenu de ce certificat avec la commande `x509` ad-hoc.

8 Vérification du certificat

Pour vérifier la validité du certificat, il faut utiliser la commande suivante, utilisant le certificat du CA qui l'a émis :

```
openssl verify -CAfile PereUbuCertif.pem monCertif.pem
```

9 Pour aller plus loin...

Les questions ci-dessous sont réservées aux curieux qui souhaitent creuser le sujet des certificats dans le cadre d'un serveur Web. Elles sont facultatives, ne constituent pas matière d'examen et ne sont pas comptabilisées dans le temps prévu pour le TP.

- *En quoi ce certificat va-t-il être utile au serveur web ?*
- *Comment faire à présent pour utiliser le certificat sur notre serveur Web ? Suppoons qu'il tourne sur Apache, quelles sont les principales étapes de configuration ?*
- *Une fois le serveur configuré avec le certificat signé, est-ce que les client Web vont l'accepter ? Que devrons-nous faire pour arranger ça ?*

10 Déliverables

Avant le début du prochain cours, vous posterez sur le Campus Virtuel un document PDF d'une page ou deux documentant le travail effectué et répondant aux questions posées.

1. Les curieux pourront aller rechercher l'origine de cet étrange mot de passe