Travaux pratiques
x509

Anaïs Gantet, Benoît Camredon

21 novembre 2019

Objectifs

- Se familiariser avec la manipulation de certificats
  - Afficher un certificat
  - Convertir un certificat dans un autre format
- Etre capable de générer des certificats
  - Certificat auto-signé
  - Demande de signature de certificat
  - Se faire une pseudo PKI
- Se familiariser avec le format PKCS12

1 Manipulation de certificats existants

L'outil `openssl x509` permet d'effectuer diverses manipulations sur les certificats.

1. De nombreux certificats se trouvent dans le répertoire `/etc/ssl/certs`. Choisir un certificat et trouver la commande `openssl x509` permettant d'afficher son contenu sous format texte.
2. Relever les différents types d'information contenus dans ce certificat.
3. Dans le répertoire `/etc/ssl/certs`, trouver deux certificats utilisant des mécanismes de signature différents. Pour chacun, relever les caractéristiques cryptographiques embarquées dans le certificat.
4. Le format de certificat le plus courant dans `/etc/ssl/certs` est le format `PEM`, mais le format `DER` reste également très répandu. Choisir un certificat `PEM` dans `/etc/ssl/certs` et trouver la commande `openssl x509` permettant le convertir en un certificat `DER`.
5. De même, convertir un certificat `DER` vers un certificat `PEM`.

2 Génération de certificats

2.1 Certificats auto-signés

1. Le type de certificats le plus simple à générer est les certificats auto-signés. En effet, il ne nécessite la mise en place d'aucune PKI particulière. Ce sont des certificats à usage interne. A l'aide de l'outil `openssl req`, choisir un nom de subject quelconque et générer un certificat auto-signé valable 10 ans, signé avec une clé RSA 4096.

2.2 CSR et CA

L'outil `openssl req` permet également de générer des demandes de signature de certificat (CSR).

1. Commencer par générer une paire de clés RSA 2048 privée/publique à l'aide de `openssl genrsa`.
2. A l'aide de `openssl req`, utiliser la paire de clé générée pour générer une nouvelle CSR, avec SHA256 comme algorithme de hachage, et popeye comme subject.
3. Afficher le contenu de la CSR sous format texte avec `openssl req` et relever les différentes informations qu'elle contient.

Cette CSR devra ensuite être envoyée à une autorité de certification (CA) pour obtenir le certificat correspondant, signé par elle. Dans le cadre du TP, nous allons nous contenter de créer une CA en local.

1. Générer une nouvelle paire de clé, mais pour la CA cette fois-ci.
2. Générer un certificat auto-signé par la CA (CN=TLSSEC). La CA possède alors tout le matériel cryptographique pour signer des CSR.
3. A l'aide de `openssl x509`, signer la CSR obtenue précédemment avec cette CA.
4. A l'aide de `openssl x509`, afficher le contenu du certificat obtenu. En particulier, vérifier que le subject est bien popeye et que l'issuer est bien TLSSEC.
5. Enfin, à l'aide `openssl verify`, vérifier que le certificat est correct.

3 Utilisation de PKCS#12

Dans certains cas, il est pratique de pouvoir distribuer le certificat et la clé privée dans un même fichier ; par exemple lors de la distribution des identifiants à nouvel utilisateur voulant se connecter à des ressources données. Le format PKCS#12 permet cela.

1. A l'aide de `openssl pkcs12`, créer un fichier PKCS#12 contenant la clé privée et le certificat de l'utilisateur popeye ayant fait la CSR à l'exercice précédent.