Université Côte d'Azur — L3 Info, L3 MI, L3 ST
Cryptographie et calcul symbolique — 2023-24

TP n° 10

Certificats – Infrastructures de Gestion de Clefs

On sait déjà utiliser OpenSSL pour faire du *chiffrement à clef secrète* et *à clef publique*, des *fonctions de hachage* et des *signatures digitales*. Aujourd'hui, nous allons explorer le maniement des *certificats* à l'aide des commandes `ca`, `req`, `x509` ou celles de la famille `pkcs`, issues de la liste suivante des *commandes standard* :

```
openssl list -commands
```

```
asn1parse       ca              ciphers         cms
crl             crl2pkcs7       dgst            dhparam
dsa             dsaparam        ec              ecparam
enc             engine          errstr          gendsa
genpkey         genrsa          help            list
nseq            ocsp            passwd          pkcs12
pkcs7           pkcs8           pkey            pkeyparam
pkeyutl         prime           rand            rehash
req             rsa             rsautl          s_client
s_server        s_time          sess_id         smime
speed           spkac           srp             storeutl
ts              verify          version         x509
```

On rappelle le pointeur sur le site officiel d'OpenSSL :

http://www.openssl.org

Certificats

Exercice 1) Ouvrez votre navigateur habituel et cherchez où se cachent les informations relatives aux *Autorités de certification-racine* dont les clefs publiques sont dans votre navigateur (ou dans votre OS) d'origine. Recherchez notamment la chaîne de certification relative à *Moodle*.

Exercice 2) Création d'un certificat auto-signé d'une AC

1. Fichier de configuration

Allez visiter le fichier de configuration `openssl.cnf` (dans `/System/Library/OpenSSL` sous MacOS ou `/etc/ssl/` sous Linux), vous comprendrez pourquoi dans la suite il faut respecter scrupuleusement les noms des répertoires et des fichiers ainsi que leurs emplacements et positions relatives. N'hésitez pas à revenir à ce fichier par la suite.

2. Création d'un fichier de caractères aléatoires

Créez un répertoire `demoCA` et un sous-répertoire `private`. Créez à l'aide de la commande UNIX `dd` un fichier aléatoire `.rand` dans le répertoire `private` en puisant dans le réservoir d'entropie qu'est le fichier `/dev/random`. Il devra contenir au moins 16 ko. Vérifiez la taille du fichier `.rand` obtenu et visualisez-le avec `xxd` ou `hexdump`.

Important : dorénavant, vous resterez positionnés **en amont de votre répertoire `demoCA`**.

A tout moment, vous pourrez vérifier l'intégralité de son contenu avec l'option -R de `ls`.

3. Mise en place d'une autorité de certification (CA)

Créez dans `demoCA` les sous-répertoires `certs`, `crl` et `newcerts`. Créez un fichier vide `demoCA/index.txt` et un autre fichier `demoCA/serial` dans lequel vous entrerez un numéro de série quelconque.

Contrôlez le tout avec une commande récursive affichant le contenu du répertoire `demoCA` tout entier. Redéfinissez *a minima* les droits d'accès à tous ces répertoires et fichiers.

(a) Avec la commande `genrsa` d'OPENSSL, créez la clef privée du CA `cakey.pem` de 4096 bits en utilisant le fichier aléatoire. Cette clef privée doit être impérativement stockée dans le répertoire `demoCA/private` et chiffrée sur le disque par AES-256. Quelle est l'utilité du fichier aléatoire ici ? Comprenez au passage ce qu'est `e` dans le message affiché `e is 65537 (0x10001)`. Vous pouvez visualiser la clef privée de l'Autorité de Certification (CA) que vous êtes !

(b) Avec la commande `req` d'OPENSSL, créez un nouveau certificat racine auto-signé, à l'aide de la clef privée précédente dont on peut déduire la clef publique. Ce certificat devra être au format `X-509`, stocké dans le répertoire `demoCA` et valable 365 jours.

(c) Utilisez la commande standard `x509` d'OPENSSL pour visualiser en clair les informations du certificat créé. Examinez son contenu.

Exercice 3) Requête en signature auprès de l'AC d'un certificat-utilisateur

1. Création de la requête par l'utilisateur

(a) En tant qu'utilisateur, vous voulez envoyer une requête de signature de certificat au CA joué par votre binôme. Vous avez besoin de générer une clef privée RSA de 1024 bits. Pour la mise en œuvre, avec de la rigueur, on peut aussi endosser les deux rôles à la fois mais c'est déconseillé ...

(b) Avec la commande `req` d'OPENSSL, créez la requête en signature du certificat de l'utilisateur auprès du CA à partir de la clef privée de l'utilisateur. Visualisez cette requête : l'en-tête confirme que ce n'est pas un certificat. Envoyez-la à votre binôme-CA pour certification.

2. Signature du certificat-utilisateur par l'AC

(a) Avec la commande `ca` d'OPENSSL, répondez en tant que CA à la requête reçue de votre binôme. Il s'agit de signer le certificat de l'utilisateur (vous choisirez éventuellement l'option `-policy policy_anything`). Notez que par défaut, le certificat s'affiche sur la sortie standard, redirigez-la. Pensez à bien regarder les informations *issuer* et *subject*. Le CA peut alors envoyer le certificat signé à l'utilisateur.

(b) Visualisez l'intégralité du répertoire `demoCA`, vous y retrouverez ce certificat. Sous quel(s) nom(s) ? Notez aussi que des fichiers de mise-à-jour ou d'incrémentation sont apparus.

3. Vérification de son certificat par l'utilisateur

(a) Vous venez de recevoir votre certificat fraîchement signé par votre binôme-CA. Visualisez-le avec la commande `x509`.

(b) Le lien est donc certifié entre vous et votre clef publique, vous n'avez pas à craindre l'attaque MITM. Vous pouvez vérifier au moyen de la commande `verify` la validité de votre propre certificat. Pour vérifier son propre certificat, un utilisateur a bien-sûr besoin du certificat auto-signé du CA-binôme, il faut le lui réclamer.

4. Révocation d'un certificat

OPENSSL permet aussi d'envisager la révocation d'un certificat. Le répertoire `crl` de `demoCA` est destiné à recevoir les certificats révoqués.

Gestion de clefs

La commande `ssh` sous LINUX lance un programme pour se connecter à une machine distante, en suivant le protocole SSH. Dans le répertoire `.ssh` à la racine du répertoire-utilisateur, ce programme maintient un fichier `known-hosts` des serveurs sur lesquels l'utilisateur s'est déjà connecté. La première connexion est considérée comme sûre c'est-à-dire qu'on considère (qu'on espère !) qu'il n'y a pas d'attaque MITM. Les suivantes sont authentifiées par la clef publique du serveur stockée dans le fichier `.ssh/known_hosts`. Ce principe TOFU aussi appelé *leap-of-faith authentication* en anglais obéit à un certain pragmatisme.

Exercice 4) `OpenSSH` sous LINUX ou MAC-OSX

1. Comment rendre cette authentification du serveur sûre ?
2. Plusieurs méthodes d'authentification du client sont alors possibles (clef publique, mot de passe, dialogue au clavier). Tentez de vous connecter à un compte tiers sur une autre machine. Comment cela se passe-t-il ?
3. Générez une paire de clefs publique/privée de type ECDSA avec l'utilitaire `ssh-keygen`.
4. Ajoutez la clef publique ainsi générée au fichier `authorized_keys` dans le répertoire `.ssh` du compte *tiers*. Connectez-vous sur ce compte en utilisant `ssh -v`. Que constatez-vous ? Effacez le contenu du fichier `authorized_keys` dans le compte-tiers.
5. Désactivez en le sauvegardant différemment le fichier `known_hosts`. Se connecter sur une autre machine. A quel message doit-on s'attendre ?
6. Modifiez le fichier `known_hosts` en changeant la clef publique et tentez à nouveau de vous connecter. A quel message doit-on s'attendre ?

Voici un exercice alternatif au cas où nous ne pourrions pas mettre en œuvre `ssh` entre les machines du P.V. :

Exercice 5) Connectez-vous sur le premier défi proposé sur :

`https://overthewire.org/wargames/bandit/bandit0.html`

et regardez le changement dans votre fichier `known_hosts`. Pour cela, effectuez :

`ssh -l bandit0 -p 2220` *hostname*

Pensez à utiliser l'option `verbose` dans ses versions `-v`, `-vv` ou `-vvv` pour comprendre quelle authentification a été utilisée.

De la cryptographie à la sécurité

Exercice 6) A l'occasion, je vous propose d'aller jeter un coup d'œil au portail *Formations* de l'ANSSI (Agence Nationale pour la Sécurité des Systèmes d'Information) :

http://www.ssi.gouv.fr/particulier/formations