Installation et utilisation des E-Token Aladdin pour les certificats robot GRID-FR

Cette documentation décrit l'installation des token Aladdin « eToken Pro 72K » sur CentOS 6.2.
Sur les autres distributions Linux, la procédure devraient être sensiblement la même. La principale limite étant la disponibilité du logiciel SafenetAuthenticationClient fourni par Safenet. Celui-ci est disponible sous forme de packages RPM ainsi que de paquets pour Debian.
Pour d'autres types de token, la différence portera sur la librairie PKCS11 spécifique qui devra être utilisée.

Pré-requis

Les logiciels suivants doivent être installés :

openssl	rpm
openssl-devel	rpm
pcsc-lite-devel	rpm
pcsc-lite-libs	rpm
libtool-ltdl	rpm
libtool-ltdl-devel	rpm
pcsc-lite	rpm
hal	rpm
ccid	rpm
opensc	http://www.opensc-project.org/opensc
libp11	http://www.opensc-project.org/libp11/
engine_pkcs11	http://www.opensc-project.org/engine_pkcs11/
SafenetAuthenticationClient	Voir le support Safenet

Récupérer le logiciel fournit par Safenet : SafeNetAuthenticationClient_Linux_8.1.zip
A l'intérieur se trouve le RPM à installer : SafenetAuthenticationClient-8.1.0-4.rpm

```
=> rpm -ivh SafenetAuthenticationClient-8.1.0-4.rpm --nodeps
```

Récupérer les logiciels libp11, engine_pkcs11 et opensc sur le site http://www.opensc-project.org/files/

compiler libp11-0.2.8

```
./configure --prefix=/usr/
make
make install
```

compiler engine_pkcs11-0.1.8

```
export LIBP11_LIBS="-L/usr/lib -lp11"
export LIBP11_CFLAGS="-I/usr/include"
```

```
./configure --prefix=/usr/
make
make install
```

compiler opensc-0.12.2

```
./configure --prefix=/usr --sysconfdir=/etc/opensc
make
make install
```

Initialisation du token

Le deamon pcscd doit être démarré.

Insérer le token USB sur la machine.
L'initialisation consiste à passer les 2 commandes suivantes :

1. Initialisation du SO PIN code (Super Officer)

```
pkcs11-tool --module <path>/libeToken.so   --init-token --label <label>
```

2. Initialisation du USER PIN code

```
pkcs11-tool --module <path>/libeToken.so --init-pin -l
```

La 1ère commande demandera d'indiquer le SO PIN nécessaire pour certaines opérations sur le token exigeant des droits supérieurs.
La 2ème commande permet de modifier le PIN code qui sera demandé pour tout accès au token. Pour cela, il faudra fournir le SO PIN initialisé précédemment.
La librairie `libeToken.so` fait partie du package fournie par Safenet.

Génération d'un bi-clé

La génération du bi-clé, nécessaire à la demande d'un certificat robot, se fait à l'aide de la commande `pkcs11-tool`. La particularité est que les clés seront générées directement sur le token.

```
pkcs11-tool --module /usr/lib64/libeToken.so -l -k --key-type
RSA:<len> --id <id> --label <label>
```

- ⇨ <len> est la longueur de la clé (1024, 2048, …)
- ⇨ <id>, l'identifiant numérique du bi-clé ainsi créé, est un nombre
- ⇨ La valeur <label> (optionnelle) est une chaine de caractère.

Par exemple :

```
$ pkcs11-tool --module /usr/lib64/libeToken.so -l -k --key-type
RSA:1024 --id 100 --label "test"
Using slot 0 with a present token (0x0)
Logging in to "Test".
Please enter User PIN: <user pin code>
```

```
Key pair generated:
Private Key Object; RSA
  label:      “test“
  ID:         100
  Usage:      decrypt, sign, unwrap
Public Key Object; RSA 1024 bits
  label:      “test“
  ID:         100
  Usage:      encrypt, verify, wrap
```

Pour afficher le contenu du token :

```
$ pkcs11-tool --module /usr/lib64/libeToken.so  -v -O -l
Using slot 0 with a present token (0x0)
Logging in to "Test".
Please enter User PIN:
Private Key Object; RSA
  label:      “test“
  ID:         100
  Usage:      decrypt, sign, unwrap
Public Key Object; RSA 1024 bits
  label:      “test“
  ID:         100
  Usage:      encrypt, verify, wrap
```

Nous retrouvons ici le bi-clé formé par la clé privée et la clé publique créées précédemment.

Demande d’un certificat

1. Créer un fichier openssl.cnf

```
openssl_conf = openssl_init
[openssl_init]
engines = engine_section
[engine_section]
pkcs11 = pkcs11_section
[pkcs11_section]
engine_id = pkcs11
dynamic_path = /usr/lib/engines/engine_pkcs11.so
MODULE_PATH = /usr/lib64/libeToken.so
init = 1

[ req ]
default_bits            = 1024
distinguished_name      = req_distinguished_name

[ req_distinguished_name ]
```

La valeur de `MODULE_PATH` indiquée ici correspond à celle fournie par Safenet pour les eToken Pro 72K. Si vous utilisez un autre type de token, il faudra indiquer la librairie PKCS11 correspondante.

2. Création de la requête PKCS10 :

```
openssl req -config openssl.cnf -engine pkcs11 -new -key <id> -
keyform engine -out req.pem -text -subj "<DN>"
```

- ⇨ La valeur de <id> doit correspondre à celle indiquée précédemment. Elle indique la clé existante dans le token à utiliser.
- ⇨ Le forme du DN doit respecter le format :

```
/O=GRID-FR/C=FR/O=<O>/OU=<OU>/CN=Robot: <nom application> - <votre
nom>/emailAddress=<emailAddressAdmin>
```

où les valeurs <O>, <OU>, et <emailAddressAdmin> doivent correspondre à celles présentes dans votre certificat personnel GRID-FR.

- ⇨ La requête PKCS10 à fournir pour votre demande de certificat robot se trouve dans le fichier `req.pem`.

3. Demander le certificat en utilisant le fichier req.pem (requête PKCS10), comme ceci :

- ⇨ aller sur le portail de demandes de certificats GRID-FR: https://pub-ee-grid.pncn.education.gouv.fr/EE/;
- ⇨ authentifier vous avec votre certificat personnel GRID-FR;
- ⇨ cliquer sur “Demander un certificat” du menu d’”Enrôlement” gauche;
- ⇨ sélectionner la catégorie d'AC “Robots”, et confirmer ensuite le profil “Robots” de certificat;
- ⇨ entrer votre adresse électronique (celle présente dans votre certificat personnel GRID-FR);
- ⇨ sélectionner le fichier PEM (requête PKCS10) déjà généré;
- ⇨ vérifier les informations affichées et puis valider.

Une fois la demande de certificat est validée par l’Autorité d’Enregistrement GRID-FR, vous recevez un mail vous indiquant comment récupérer le certificat.

4. Récupérer le certificat en format DER

Ou vous pouvez récupérer le certificat en format PEM et le transformez ensuite en DER

```
openssl x509 –in cert.pem –out cert.der -outform DER
```

5. Insertion du certificat dans le token

```
pkcs11-tool --module /usr/lib64/libeToken.so  -w cert.der -l --label
“robot test cg“ --id <id> -y cert
```

L’affichage du contenu du token indiquera :

```
$ pkcs11-tool --module /usr/lib64/libeToken.so  -v -O -l
Using slot 0 with a present token (0x0)
Logging in to "Test".
Please enter User PIN:
Private Key Object; RSA
```

```
  label:      “test“
  ID:         100
  Usage:      decrypt, sign, unwrap
Public Key Object; RSA 1024 bits
  label:      “test“
  ID:         100
  Usage:      encrypt, verify, wrap
Certificate Object, type = X.509 cert
  label:      robot test cg
  ID:         100
```

Le token contient maintenant la clé privée, la clé publique et le certificat associé.