Configuration de CAS et uPortal avec des certificats de la PKI pilote du CRU

On suppose ici que le serveur CAS est porté sur la machine cas.univ-xxx.fr, et uportal sur la machine uportal.univ-xxx.fr

On dispose des fichiers suivants, générés par le CRU (stockés dans /etc/x509) :

- cachain.pem : chaine d'autorités de CA (ac-racine -> ac-serveur)
- ac-serveur.pem : c'est le certificat ac-serveur du CRU
- cas.univ-xxx.fr.crt : le certificat pour la machine cas (signé par ac-serveur)
- cas.univ-xxx.fr.key : la clé privée pour la machine cas
- uportal.univ-xxx.fr.cer : le certificat pour la machine uportal (signé par ac-serveur)
- uportal.univ-xxx.fr.key : la clé privée pour la machine uportal
  Est décrite ici l'installation du certificat sur la machine cas sans frontal apache. La procédure est similaire pour la machine uportal.

> **Important**
>
> Il est plus facile de travailler avec un frontal apache, cf DOC:Pourquoi+utiliser+des+certificats+pour+CAS.
>
> Sinon, il est possible, avec la PKI du CRU, de travailler différemment ; c'est plus simple dans un environnement java.
> Il suffit, à l'aide de keytool, de générer la bi-clé et une requête de certification.
> Il faut alors d'envoyer cette requête de certification au CRU, qui va nous retourner le certificat X509 correspondant.
> Toute la procédure pour générer le keystore peut alors se faire uniquement avec l'utilitaire keytools, sans avoir besoin d'autres outils.
> Voir : Utilisation de certificats X509 en Java

Création du keystore pour le serveur CAS

Il faut au préalable récupérer le package org.mortbay.jetty-jdk1.2.jar livré avec la distribution de Jetty.

On va ensuite procéder en plusieurs étapes :

- génération d'un certif pkcs12 depuis la clé privée du serveur et son certif
- génération du keystore depuis le pkcs12
- ajout de la chaine de certification
  Si on n'ajoute pas la chaine de certification, celle-ci n'est pas transmise aux navigateurs clients, ce qui peut poser problème.

Récupération/installation de Jetty

Récupérer Jetty-x.y.z.tgz de http://jetty.mortbay.org (dernière version 4.2.9) et installer :

```
% cd /usr/local
% mv /tmp/Jetty-4.2.9.tgz src
% tzxf src/Jetty-4.2.9.tgz
```

Le package contient l'utilitaire PKCS12Import qui sait créer un fichier keystore à partir d'un certificat pkcs12.

Génération du certificat serveur au format pkcs12

On génère le certif pkcs12 depuis la clé privée du serveur et son certificat PEM (on entre le mot de passe xxxxxx en interactif) :

```
% cd /etc/x509
% openssl pkcs12 -export -inkey cas.univ-xxx.fr.key -in cas.univ-xxx.fr.cert -out cas.pkcs12
Enter Export Password: xxxxxx
Verifying - Enter Export Password: xxxxxx
```

On peut le contrôler (keytool sait lire un fichier pkcs12) :

```
% keytool -list -v -keystore cas.pkcs12 -storetype pkcs12 -storepass xxxxxx
Type Keystore : pkcs12
Fournisseur Keystore : SunJSSE

Votre Keystore contient 1 entrée(s)

Nom dalias : {0}
Date de création : 23 juin 2003
Type d'entrée : keyEntry
Longueur de chaîne du certificat : 1
Certificat[1]:
Propriétaire : EMAILADDRESS=reseau@univ-xxx.fr, CN=cas.univ-nancy2.fr, O=Universite xxx, C=FR
Émetteur : EMAILADDRESS=ca-admin@cru.fr, CN=ac-serveur, O=CRU, C=FR
Numéro de série : 6
Valide du : Wed Jun 04 17:13:25 CEST 2003 au : Fri Jun 04 17:13:25 CEST 2004
Empreintes de certificat :
         MD5 :  09:10:0D:03:77:A6:D2:2D:DA:92:7D:65:E8:A3:1E:90
         SHA1: C5:55:94:B1:39:8A:67:09:1B:E4:F0:C4:FE:2D:31:96:8F:9C:31:AC
```

Génération du magasin de clé

C'est le keystore qui va contenir clés privée et publique, et le certificat ; c'est celui qui devra être déclaré dans le fichier server.xml du serveur tomcat de CAS.

On utilise l'utilitaire PKCS12Import (on donne un mot de passe yyyyyy au nouveau magasin) :

```
% java -classpath /usr/local/Jetty-4.2.9/lib/org.mortbay.jetty-jdk1.2.jar org.mortbay.util.PKCS12Import cas.
pkcs12 cas.keystore
Enter input keystore passphrase: xxxxxx
Enter output keystore passphrase: yyyyyy
Alias 0: 1
Adding key for alias 1
```

Le nom d'alias créé est 1. Si on veut le changer (en cas, par exemple) :

```
% keytool -keyclone -keystore cas.keystore -alias 1 -dest cas -storepass yyyyyy
Spécifiez le mot de passe de la clé pour <cridev>
        (appuyez sur Entrée si le résultat est identique à <1>)
```

Contrôle (le magasin doit maintenant contenir les 2 alias) :

```
% keytool -list -keystore cas.keystore -storepass yyyyyy
Type Keystore : jks
Fournisseur Keystore : SUN

Votre Keystore contient 2 entrée(s)

1, 23 juin 2003, keyEntry,
Empreinte du certificat (MD5) : 09:10:0D:03:77:A6:D2:2D:DA:92:7D:65:E8:A3:1E:90
cas, 23 juin 2003, keyEntry,
Empreinte du certificat (MD5) : 09:10:0D:03:77:A6:D2:2D:DA:92:7D:65:E8:A3:1E:90
```

On peut ensuite effacer le premier :

```
% keytool -delete -alias 1 -keystore cas.keystore -storepass yyyyyy
% keytool -list -keystore cas.keystore -storepass yyyyyy
Type Keystore : jks
Fournisseur Keystore : SUN

Votre Keystore contient 1 entrée(s)

cridev, 23 juin 2003, keyEntry,
Empreinte du certificat (MD5) : 09:10:0D:03:77:A6:D2:2D:DA:92:7D:65:E8:A3:1E:90
```

Ajout de la chaine d'autorités de certification

Récupérer le fichier cachain.txt contenant les certificats des AC ac-racine et ac-serveur (donc, la chaine de certification) à http://igc.cru.fr/ac-serveur/, choisir 'Chaine de certification).

Pour cela, on concatène le fichier cas.univ-xxx.crt avec cachain.pem (le echo sert à ajouter un CR au fichier cas.univ-xxx.crt) :

```
% (cat cas.univ-xxx.crt ; echo ; cat cachain.txt) > certchain-cas.pem
```

keytool n'aime pas ce format de fichier. 2 solutions, le transformer en format DER avec openssl, ou plus simple, éditer ce fichier pour éliminer tout ce qui n'est pas entre des lignes ~~-----BEGIN CERTIFICATE-----~~ et ~~-----END CERTIFICATE-----~~
On choisit cette seconde méthode, ce qui donne quelque chose du genre :

```
----BEGIN CERTIFICATE-----
MIIEBzCCAu+gAwIBAgICANcwDQYJKoZIhvcNAQEEBQAwUDELMAkGA1UEBhMCRlIx
... certif de serv.univ.fr ...
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIID1zCCAr+gAwIBAgIBAzANBgkqhkiG9w0BAQQFADBPMQswCQYDVQQGEwJGUjEM
... certif de l'ac-serveur ...
nbnQ04fo+LDLFWa9qCI8mLX1JGcP4WDKJ9hk
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIID1jCCAr6gAwIBAgIBADANBgkqhkiG9w0BAQQFADBPMQswCQYDVQQGEwJGUjEM
... certif de l'ac-racine ...
6cKi0FjWaBYjBOH6/ULalf1g4P38vbUAT4A=
-----END CERTIFICATE-----
```

et on importe dans le keystore :

```
% keytool -import -alias cas -trustcacerts -file certchain-cas.pem -keystore cas.keystore -storepass yyyyyy
Certificat du plus haut niveau dans la réponse :

Propriétaire : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
Émetteur : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
Numéro de série : 0
Valide du : Thu Apr 17 16:16:24 CEST 2003 au : Sun Apr 14 16:16:24 CEST 2013
Empreintes de certificat :
         MD5 :  25:44:35:56:18:E4:EF:65:02:25:99:C6:4E:17:00:38
         SHA1: 33:88:49:67:76:43:8F:58:4D:1F:93:9B:55:CF:C3:A5:37:23:39:B4

... n'est pas digne de confiance. Installer la réponse quand même ? [non] :  oui
Réponse de certificat installée dans le Keystore
```

Contrôle (cas.keystore doit contenir une entrée de type keyEntry, une longueur de chaine de certif de 3, et les 3 certificats de la chaîne de confiance) :

```
% keytool -list -v -keystore cas.keystore
Type Keystore : jks
Fournisseur Keystore : SUN

Votre Keystore contient 1 entrée(s)

Nom dalias : {0}
   Date de création : 23 juin 2003
   Type d'entrée : keyEntry
   Longueur de chaîne du certificat : 3
   Certificat[1]:
   Propriétaire : EMAILADDRESS=reseau@univ-nancy2.fr, CN=cridev.univ-nancy2.fr,    O=Universite de Nancy 2, C=FR
   Émetteur : EMAILADDRESS=ca-admin@cru.fr, CN=ac-serveur, O=CRU, C=FR
   Numéro de série : 6
   Valide du : Wed Jun 04 17:13:25 CEST 2003 au : Fri Jun 04 17:13:25 CEST 2004
   Empreintes de certificat :
   MD5 : 09:10:0D:03:77:A6:D2:2D:DA:92:7D:65:E8:A3:1E:90
   SHA1: C5:55:94:B1:39:8A:67:09:1B:E4:F0:C4:FE:2D:31:96:8F:9C:31:AC
   Certificat[2]:
   Propriétaire : EMAILADDRESS=ca-admin@cru.fr, CN=ac-serveur, O=CRU, C=FR
   Émetteur : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
   Numéro de série : 3
   Valide du : Fri May 16 11:39:11 CEST 2003 au : Fri Dec 14 10:39:11 CET 2012
   Empreintes de certificat :
   MD5 : 1D:51:5F:45:D1:CC:E9:C4:DA:4A:F6:F6:F9:CD:E3:11
   SHA1: 4E:27:8C:05:B5:6C:03:EA:A1:5B:F0:01:AE:D1:86:43:E9:03:6D:B6
   Certificat[3]:
   Propriétaire : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
   Émetteur : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
   Numéro de série : 0
   Valide du : Thu Apr 17 16:16:24 CEST 2003 au : Sun Apr 14 16:16:24 CEST 2013
   Empreintes de certificat :
   MD5 : 25:44:35:56:18:E4:EF:65:02:25:99:C6:4E:17:00:38
   SHA1: 33:88:49:67:76:43:8F:58:4D:1F:93:9B:55:CF:C3:A5:37:23:39:B4
```

Supprimer les fichiers intermédiaires

```
/bin/rm certchain-cas.pem cas.pkcs12
```

Paramétrer tomcat pour https

La procédure est la même que pour l'installation rapide. Dans le fichier server.xml de tomcat (5.5 ou 6) :

```
<Connector port="8443" maxHttpHeaderSize="8192"
                       maxThreads="2000" minSpareThreads="25" maxSpareThreads="75"
                       enableLookups="false" disableUploadTimeout="true"
                       acceptCount="200" scheme="https" secure="true" SSLEnabled="true"
                        clientAuth="false" sslProtocol="TLS"
                        keystoreFile="/etc/x509/cas.keystore" keystorePass="yyyyyyy" />
```

Tester en accédant https://cas.univ-xxx.fr:8443.

Prise en compte de l'autorité de CA

Les paragraphes précédents ont permis de faire fonctionner les 2 serveurs tomcat (uportal et cas) en HTTPS.

Ces 2 serveurs vont avoir besoin de dialoguer directement entre eux en HTTPS. Il faut donc qu'ils puissent valider mutuellement le certificat de l'autre, sinon :

```
ERROR - YaleCASContext: javax.net.ssl.SSLHandshakeException: java.security.cert.CertificateException: Couldn't
find trusted certificate
```

Pour celà, on va déclarer l'autorité de CA ac-racine dans le keystore de la machine virtuelle java.

Transformation du format du certificat de l'autorité racine

L'utilitaire keytool veut du format DER (en fait, il peut utiliser du format pem, mais expurgé du 'superflu' : il ne faut laisser que les infos entre 'BEGIN CERTIFICATE' et 'END CERTIFICATE'). On transforme donc les certificats comme ceci :

```
% openssl x509 -in ac-racine.pem -out ac-racine.der -outform DER
```

Pour s'assurer de la lisibilité du certificat :

```
% keytool -printcert -file ac-racine.der
Propriétaire : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
Émetteur : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
Numéro de série : 0
Valide du : Thu Apr 17 16:16:24 CEST 2003 au : Sun Apr 14 16:16:24 CEST 2013
Empreintes de certificat :
         MD5 :  25:44:35:56:18:E4:EF:65:02:25:99:C6:4E:17:00:38
         SHA1: 33:88:49:67:76:43:8F:58:4D:1F:93:9B:55:CF:C3:A5:37:23:39:B4
```

Intégration de l'autorité de certification du CRU dans Java

2 méthodes :

- soit on enregistre l'AC du CRU dans le keystore global de la JVM
- soit on l'enregistre dans un keystore dédié à une instance de JVM.

Les 2 solutions sont présentées.

Intégration de l'autorité de certification du CRU dans la JVM

% keytool -import -alias CRU-ac-racine -file ac-racine.der -keystore $JAVA_HOME/jre/lib/security/cacerts -storepass changeit
Propriétaire : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
Émetteur : EMAILADDRESS=ca-admin@cru.fr, CN=ac-racine, O=CRU, C=FR
Numéro de série : 0
Valide du : Thu Apr 17 16:16:24 CEST 2003 au : Sun Apr 14 16:16:24 CEST 2013
Empreintes de certificat :
MD5 : 25:44:35:56:18:E4:EF:65:02:25:99:C6:4E:17:00:38
SHA1: 33:88:49:67:76:43:8F:58:4D:1F:93:9B:55:CF:C3:A5:37:23:39:B4
Faire confiance à ce certificat ? non : oui
Certificat ajouté au Keystore
Contrôle :

```
% keytool -list -keystore $JAVA_HOME/jre/lib/security/cacerts -storepass changeit
Votre keystore contient 17 entrée(s)
Type Keystore : jks
Fournisseur Keystore : SUN

[...]
cru-ac-racine, 23 juin 2003, trustedCertEntry,
Empreinte du certificat (MD5) : 25:44:35:56:18:E4:EF:65:02:25:99:C6:4E:17:00:38
```

Intégration de l'autorité de certification du CRU dans un keystore 'privé'

L'objectif est de faire en sorte que le serveur de servlet (en l'occurence, tomcat) utilise un keystore qui lui est dédié pour les autorités de certification 'trusted', au lieu du keystore de la JVM.

Création du kestore :

```
keytool -import -v -trustcacerts -alias CRU-ac-racine -file ac-racine.der -keystore cru-ac-racine.cert -
storepass yyyyyy
```

Maintenant, lors du lancement de tomcat qui doit faire confiance à CAS :

```
CATALINA_OPTS="-Djavax.net.ssl.trustStore=cru-ac-racine.cert";export CATALINA_OPTS
```