Installation des certificats esup-smsu 1.x

Documentation pour l'ancienne version de SMS-U

Mise en place de l'authentification mutuelle entre front office et back office

Principes – définitions

? Pièce jointe inconnue

Le front office et le back office communiquent via SSL en utilisant des certificats x509.

L'authentification est dite « mutuelle ». Le front office et le back office doivent chacun avoir un certificat propre qu'ils s'échangent lors de chaque connexion l'un à l'autre. Lors de la mise en place de la connexion front->back, chaque extrémité envoi à l'autre son certificat. L'extrémité doit alors valider ce certificat.

Si le certificat reçu est issu d'une autorité de certification, des mécanismes automatiques sont déclenchés pour valider le certificat.

Si le certificat reçu est auto-signé ou bien si le serveur dans l'incapacité d'utiliser les mécanismes automatiques de validation, le serveur va regarder dans une liste de certificats de confiance s'il y trouve ce certificat.

Keystore et truststore

Afin d'effectuer cette authentification mutuelle deux notions sont importantes sur chaque serveur tomcat/java : le keystore et le truststore.

- **Keystore** : chaque serveur (front et back office) doit posséder un keystore. Le keystore contient des ensembles clé publique/clé privée/certificat. Ces éléments sont ses identifiants.
- **Truststore** : chaque serveur (front et back office) doit posséder un truststore. Le truststore contient la liste de certificats de confiance.

Au sein d'un keystore ou d'un truststore, chaque certificat (et clés associées dans le cadre des keystore) est identifié par un alias.

Paramétrage serveur back office

Le back office est déployé en tant que servlet et fourni des webservices.
La mise en place de la liaison sécurisée avec authentification du client se fait de manière assez classique :

- soit au niveau du paramétrage du frontal apache si un frontal apache est utilisé
- sinon au niveau du paramétrage du connecteur du serveur tomcat utilisé

Même pour les tests, il est conseillé d'utiliser un certificat issu d'une autorité de certification pour le serveur back office.
Il est aussi possible d'utiliser un certificat auto-signés, mais alors le truststore de chaque **front office** doit posséder le certificat du back office.

Avec frontal apache

Configurer le <VirtualHost> avec les paramètres suivant (adapter les noms de fichiers à vos besoins) :

```
SSLEngine on
SSLCertificateFile /etc/ssl/certs/sms.univ-xxx.fr.crt
SSLCertificateKeyFile /etc/ssl/private/sms.univ-xxx.fr.key
SSLCertificateChainFile /etc/ssl/certs/cachain.crt

SSLCACertificateFile /etc/ssl/certs/ca.crt
SSLCACertificatePath /etc/ssl/certs/smsuapi.crt
SSLVerifyClient optional
SSLVerifyDepth 3
```

Créer /etc/ssl/certs/smsuapi.crt/ et le remplir au fur à mesure avec les certificats des front offices s'ils sont auto-signés :

```
mkdir /etc/ssl/certs/smsuapi.crt
```

- sur un serveur front office :

  ```
  keytool -export -rfc -keystore keystore-smsu.jks -alias smsu -file le_CN.crt
  ```

- sur le serveur back office

  ```
  cp le_CN.crt /etc/ssl/certs/smsuapi.crt/
  c_rehash /etc/ssl/certs/smsuapi.crt
  ```

où le_CN est le nom du CN choisi lors de la création du certificat front office auto-signé

Sans frontal apache

Génération du keystore back office

à partir de certificats au format apache

Cf Utilisation de certificats X509 en Java#Importation d'une clé privée

création d'une clé et la signer avec une autorité de certification

Cf Utilisation de certificats X509 en Java#Signer les certificats par une AC locale ou Utilisation de certificats X509 en Java#Faire signer un certificat par une autorité de certification externe

auto-signé

Dans tous les web services mis en place, les front offices sont des clients (ils initient la connexion) et le back office serveur. Le certificat d'un serveur doit respecter une règle :

> *le CN (Common Name, correspondant à la question nom et prénom lors de la génération par keytool) doit correspondre au nom de la machine.*

Exécutez la commande suivante (nb : validité de 3 ans) :

```
keytool -genkey -keyalg RSA -alias smsuapi -keystore keystore-smsuapi.jks -validity 1095
```

Répondez aux questions posées en respectant la règle énoncée ci-dessus.

Voir aussi Utilisation de certificats X509 en Java#Génération d'un certificat auto-signé RSA

Configuration du truststore

Le truststore par défaut $JAVA_HOME/jre/lib/security/cacerts est une bonne base et permettra aux front office ayant un certificat signé par une autorité de certification de se connecter.

```
cp $JAVA_HOME/jre/lib/security/cacerts truststore-smsuapi.jks
```

La manipulation suivante est à effectuer pour extraire le certificat d'un keystore front office auto-signé et l'ajouter au truststore du back office :

- sur le serveur front office

  ```
  keytool -export -keystore keystore-smsu.jks -alias smsu -file smsu.crt
  ```

- sur le serveur back office

  ```
  keytool -import -keystore truststore-smsuapi.jks -alias smsu -file smsu.crt
  ```

Pour plus d'information sur ces commandes, voir Utilisation de certificats X509 en Java

NB : le certificat smsu.crt est aussi un élément de paramétrage de l'application dans l'administration du back office.

Paramétrage de tomcat

en mode quick start

Lors d'un déploiement en mode quick start, le paramétrage des keystore et truststore back office s'effectue via les propriétés suivantes du fichier **build.properties** :

- tomcat.keystore : chemin vers le keystore
- tomcat.keypass : mot de passe du keystore
- tomcat.truststore : chemin vers le truststore
- tomcat.truststorePass : mot de passe du truststore

Le fichier **server.xml** utilisé par le serveur tomcat lancé est celui situé dans le dossier **utils/tomcat**. Le connecteur est paramétré comme suit :

```
<Connector
    port="${tomcat.port}"
    maxHttpHeaderSize="8192"
    maxThreads="150"
    minSpareThreads="25"
    maxSpareThreads="75"
    enableLookups="false"
    acceptCount="100"
    connectionTimeout="20000"
    disableUploadTimeout="true"
    scheme="https" secure="true" SSLEnabled="true"
    clientAuth="optional" sslProtocol="TLS"
    keystoreFile="${tomcat.keystore}"
    keystorePass="${tomcat.keypass}"
    truststoreFile="${tomcat.truststore}"
    truststorePass="${tomcat.truststorePass}"
/>
```

en production

De manière similaire à la configuration server.xml ci-dessus, il faut préciser les paramètres keystoreFile, keystorePass, truststoreFile, truststorePass du <Connector> https dans **server.xml**

Paramétrage front office

Génération d'un keystore front office auto-signé

Les keystore front office contiennent les certificats qui serviront à authentifier les applications appelantes au sein des back office.
Chaque certificat front office doit respecter la règle suivante :

> *le CN (Common Name, correspondant à la question nom et prénom lors de la génération par keytool) doit correspondre au nom de l'application qui sera paramétrée dans le back office.*

Exécutez la commande suivante (nb : validité de 3 ans) :

```
keytool -genkey -keyalg RSA -alias smsu -keystore keystore-smsu.jks -validity 1095
```

Répondez aux questions posées en respectant la règle énoncée ci-dessus.

Si un keystore existe déjà dans l'environnement, il faut lui ajouter keystore-smsu.jks avec la commande keytool -importkeystore (nécessite java 6)

Configuration du truststore

Un truststore contient tous les certificats de confiance d'une machine.
Un truststore existe déjà et contient les CA standards, notamment "AddTrust External CA Root" pour TERENA.

en cas de certificat back office signé par une autorité de certification

Il n'y a rien à faire, le truststore existant est suffisant

en cas de certificat back office auto-signé

Il ne faut surtout pas supprimer les certificats existants du truststore.
Il faut ajouter le certificat du back office au truststore existant.

- Créez truststore-smsu.jks à partir du truststore existant (esup-portail.keystore ou $JAVA_HOME/jre/lib/security/cacerts)
- Pour chaque truststore front office, il faut effectuer les manipulations suivantes pour que chaque truststore contienne le certificat du back office.
  - extraire le certificat du keystore back office

    ```
    keytool -export -keystore keystore-smsuapi.jks -alias smsuapi -file smsuapi.crt
    ```

  - l'ajouter au truststore front office :

    ```
    keytool -import -keystore truststore-smsu.jks -alias smsuapi -file smsuapi.crt
    ```

    Pour plus d'information sur ces commandes, voir Utilisation de certificats X509 en Java

Déploiement servlet

Afin de faciliter la mise en place lors d'un déploiement servlet du front office, la mise en place des keystore et truststores (qui sont uniques pour un environnement) a été mise en place au sein même de l'application.

Les propriétés suivantes du fichier **config.properties** sont à renseigner :

- smsuapi.ws.trustStore : chemin vers le truststore
- smsuapi.ws.trustStorePassword : mot de passe du truststore
- smsuapi.ws.keyStore : chemin vers le keystore
- smsuapi.ws.keyStorePassword : mot de passe du keystore

Ensuite, décommentez le bean **initSslParameters** du fichier **properties/client/client.xml**

Déploiement portlet

Dans le cadre d'un déploiement portlet, keystore et truststore sont fournis par le portail. La mise en place des certificats du service SMS-U doit venir compléter l'existant (notamment utilisé pour l'authentification au serveur CAS). Les clés et certificats doivent être importés dans les keystore et truststore existants et être paramétrés pour être pris en compte au lancement du portail.

Une solution consiste à modifier le fichier env.cmd du portail de la sorte :

```
SET TRUST_CERT=[chemin du truststore]
SET KEY_CERT=[chemin du keytore]/keystore-smsu.jks
SET PWD_KEYSTORE=[mot de passe du keystore]
```

Puis modifier le fichier start-esup pour y intégrer la commande suivante :

```
SET CATALINA_OPTS="-Djavax.net.ssl.keyStore=%KEY_CERT%" "-Djavax.net.ssl.keyStorePassword=%PWD_KEYSTORE%" "-
Djavax.net.ssl.trustStore=%TRUST_CERT%"
```

Debuggage ssl

La sécurisation des connexions est un point sensible de l'environnement.

Tester la configuration avec openssl s_client

```
openssl s_client -host smsubackend -port 443
openssl s_client -host smsubackend -port 443 -cert smsu.crt -key smsu.key
```

Débugger

Il est possible de suivre les événements SSL de l'application en paramétrant le variable d'environnement suivante avant lancement du serveur Tomcat :

```
CATALINA_OPTS = "-Djavax.net.debug=ssl"
```

Glossaire

Glossaire des manuels du service SMS-U