TD12 Sockets SSL

Exercice 1 : sockets sécurisées

Recommandation : lire ce cours en ligne

`http://www.churchillobjects.com/c/11201.html`

Vous disposez comme point de départ des 3 classes suivantes :

```
SecureServer
CommonSocketFactory
SecureClient
```

Question 1.1 : Avec des certificats auto-signés

1. exécuter `SecureServer`, à partir de `dumbledore` par exemple et `SecureClient`, à partir de votre machine locale. Dans un premier le temps, les sockets créées sont classiques, i.e. non-cryptées. Vérifier que tout se passe normalement : le message "Salut" doit parvenir au serveur. Avec des outils spécialisés comme `Ethereal`, on peut observer les paquets qui transitent : ainsi dans un des paquets TCP, on peut visualiser le message "Salut"...
2. basculer les sockets en mode SSL (en positionnant le booléen à vrai, côté client et côté serveur).
3. la création de la socket SSL serveur échoue, car il manque le fichier (magasin) `server_keystore` permettant le stockage des clefs et des hotes authentifies.
4. créer le magasin :
   ```
   keytool -genkey -alias SecureServer -keyalg RSA -keystore server_keystore
           -dname "cn=SecureServer" -keypass master1 -storepass master1
   ```
   et vérifier que la création de socket SSL serveur réussit.
5. à partir d'une autre machine, exécuter `SecureClient`, en précisant dans le code que le serveur est `dumbledore` : la création de la socket SSL échoue, car il manque le fichier (magasin) `client_keystore` permettant le stockage des clefs et des hotes authentifies.
6. créer le magasin :
   ```
   keytool -genkey -alias SecureClient -keyalg RSA -keystore client_keystore
           -dname "cn=SecureClient" -keypass master1 -storepass master1
   ```
7. la création des sockets réussit maintenant : pourtant, nous n'avons pas encore enregistré le client comme authentique auprès du serveur (et vice-versa) ! De fait, l'absence d'authentification ne bloque pas la création de la connexion, par contre les communications sont impossibles (pas d'accès aux flux des sockets...)
8. avant d'enregistrer le client comme authentique auprès du serveur, regarder la liste des clefs dans le magasin du serveur :
   ```
   keytool -list -keystore server_keystore
   ```
9. auo-signer le certificat du client
   ```
   keytool -selfcert -alias SecureClient
            -keystore client_keystore -keypass master1 -storepass master1
   ```
10. exporter le certificat dans un fichier `client_cert` et le transmettre (par ex. avec `scp` au serveur
    ```
    keytool -export -alias SecureClient -file client_cert
           -keystore client_keystore -storepass master1
    ```
11. importer le certificat du client dans le magasin du serveur :

```
keytool -import -alias SecureClient -file client_cert
        -keystore server_keystore -storepass master1
```

et vérifier qu'il a bien été ajouté :

```
keytool -list -keystore server_keystore -storepass master1
```

Observer que le certificat du client est marqué comme sûr (`TrustedCertEntry`).

12. la communication ne marche toujours pas, car nous avons authentifié le client auprès du serveur mais pas le serveur auprès du client : reprendre les points ci-dessus en inversant client/serveur.
13. Cela marche maintenant (ouf !) : avec `Ethereal`, on ne voit plus passer de trame avec en clair "Hello".

Question 1.2 : Avec votre propre serveur de certification

Effacer au préalable les magasins !

serveur cert créer l'arborescence et placez vous dans `makecert`

```
mkdir makecert
mkdir makecert/demoCA
mkdir makecert/demoCA/certs
mkdir makecert/demoCA/crl
mkdir makecert/demoCA/newcerts
mkdir makecert/demoCA/private
touch makecert/demoCA/index.txt
echo "01" > makecert/demoCA/serial
```

serveur cert génération de la clef du serveur de certification

```
openssl genrsa -out ca.key 1024
```

serveur cert création du certificat signé

```
openssl req -new -x509 -key ca.key -out demoCA/cacert.pem
```

client création de la clef pour le client (! l'entrée cn doit correspondre à celle qui a été utilisée pour le certificat signé du serveur de certification)

```
keytool -genkey -alias SecureClient -keystore client_keystore
```

client exporter la clef publique du client du magasin pour une demande de certificat

```
keytool -keystore client_keystore -certreq -alias SecureClient
        -file clientapp.crs -storepass master1
```

serveur cert signer la clef publique du client avec la clef privée CA du serveur de certification

```
openssl ca -in clientapp.crs -out clientapp.pem -keyfile ca.key
```

serveur cert convertir la clef au format der

```
openssl x509 -in clientapp.pem -out clientapp.der -outform DER
```

client récupérer `demoCA/cacert.pem` et `clientapp.der`, et enregistrer le certificat du serveur de certification et la clef certifiée dans le magasin du client :

```
keytool -keystore client_keystore -alias serveur_cert -import -file cacert.pem
keytool -keystore client_keystore -alias SecureClient -import -file clientapp.der
```

et vérifier la présence des deux certificats dans le magasin

1. faire la même chose pour le serveur (ne pas mettre exactement les mêmes informations !) et vérifier que la communication marche bien ...