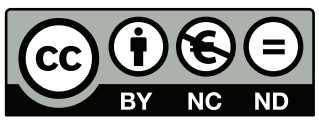
KeyTool

Key and Certificate Management Tool

Didier BERNAUDEAU

http://didier.bernaudeau.net/slide

4 février 2015



Attribution-NonCommercial-NoDerivatives 4.0 International

Sommaire

Introduction

Présentation de Keytool

- Keytool est un gestionnaire de magasin de certificat (Clés cryptographiques et certificats)
- Keytool est un outil en ligne de commande
- Keytool est édité par Oracle

Obtenir KeyTool

- Télécharger JRE (Java Runtime Environment) ou JDK (Java Development Kit) sur le site de l'éditeur (Oracle)
- Le fichier exécutable se trouve dans le dossier $JAVA_HOME/bin

Sommaire

Les fonctionnalités de keytool (1/2)

Création

- -genkeypair : créer une bi-clé et le certificat associé
- -genseckey : créér une clé secrète

Importer

- -importcert : importer un certificat au format X509
- -importkeystore : importer un magasin de certificats

Exporter

- -certreq : créer une demande de certificat
- -exportcert : exporter un certificat

Les fonctionnalités de keytool (2/2)

Afficher

- -list : afficher les éléments d'un magasin de certificats (Les clés ne seront jamais affichées)
- -printcert : afficher un certificat

Gérer

- -storepasswd : changer le mot de passe du magasin de certificats
- -keypasswd : changer le mot de passe d'une clé
- -delete : supprimer un élément du magasin de certificats
- -changealias : modifier l'alias d'un élément du magasin de certificats

Sommaire

Les formats de magasin de clés

Avec une installation standard du JRE/JDK, KeyTool supporte 3 formats :

- JKS (Java KeyStore)
- JCEKS (Java Cryptography Extension KeyStore)
- PKCS#12 (Public Key Cryptographic Standards #12)

Formats supplémentaires

Par défaut, JRE/JDK dispose des CSP (Cryptographic Service Provider) permettant de supporter les 3 formats ci-dessus.
Un CSP est une bibliothèque proposant des fonctions cryptographiques. Pour supporter d'autres formats, il est possible d'installer des CSP supplémentaires comme, par exemple, BouncyCastle (Format BKS).

Les formats de magasin de clés

JKS (Java KeyStore)

Caractéristiques

- Format défini par Oracle
- Format disponible depuis Java 1.1 (1997)
- Format pris en charge pas le CSP Sun fourni avec le JRE/JDK
- Format permettant de stocker des bi-clés et des certificats

Faible protection des clés

Une empreinte (SHA) est calculée à partir du mot de passe et une valeur alétaoire. Le mot de passe "chiffré" est le résultat d'un XOR entre l'empreinte et la clé privée.
Pour plus de détail, consulter la classe "KeyProtector" du package "sun.security.provider".

Restriction cryptographique

Ce format a été conçu pour respecter l'ancienne réglementation des Etats-Unis concernant l'exportation des outils de chiffrement.

Les formats de magasin de clés

JCEKS (Java Cryptography Extension KeyStore)

Caractéristiques

- Format défini par Oracle
- Format disponible depuis Java 5 (2002)
- Format pris en charge par le CSP SunJCE fourni avec le JRE/JDK
- Format permettant de stocker des clés secrètes, des bi-clés et des certificats

Protection des clés renforcées

Les clés sont chiffrées avec les algorithmes suivants : PBE (Password-Based Encryption) avec MD5 et DES3 en mode CBC. Pour plus de détail, consulter la classe "KeyProtector" du package "sun.crypto.provider".

Les formats de magasin de clés

PKCS12 (Public Key Cryptographic Standards #12)

Caractéristiques

- Format défini par RSA Security
- Format disponible depuis Java 5 avec le CSP SunJSSE fourni avec le JRE/JDK
- Ce format permet de stocker des bi-clés et des certificats

Support limité

Avec le CSP SunJSSE fourni en standard avec le JRE/JDK, ce format n'est supporté qu'en lecture seule. KeyTool étant lui même développé en Java, il pourra seulement lire ce format.

Sommaire

Les types de magasin de clés

TrustStore

- Contient uniquement des certificats électroniques des autorités de certification
- Nécessaire pour une application devant initier une connexion SSL SA (Simple Authentication) vers un serveur

KeyStore

- Contient des certificats électroniques et des clés cryptographiques (clés privées et/ou secrètes)
- Nécessaire pour une application devant initier une connexion SSL MA (Mutual Authentication) vers un serveur

Sommaire

Protéger le magasin de clé

- Définir le propriétaire du fichier et limiter l'élévation de privilège vers cet utilisateur
- Restreindre l'accès au fichier en lecture seule uniquement pour le propriétaire du fichier
- L'agent de sauvegarde ne doit pas sauvegarder le magasin de clé (Si besoin, le fichier doit faire l'objet d'un séquestre)
- Signer le fichier (option -storepass) et chiffrer les clés cryptographiques (option -keypass)

Sommaire

Afficher un certificat

```
$ keytool -printcert -file ca.crt
```

Paramètres

- -file : spécifie le nom du fichier du certificat

Sommaire

Afficher le contenu du magasin de clés du JRE/JDK

```
$ keytool -list -storetype JKS -keystore $JAVA_HOME/jre/lib/security/cacerts \
  -storepass changeit -v
```

Paramètres

- -storetype : spécifie le type de magasin de clés
- -keystore : spécifie le nom du fichier du magasin de clés
- -storepass : spécifie le mot de passe du magasin de clés pour vérifier l'intégrité
- -v : affiche le certificat sous la forme de caractères

Magasin de clés du JRE/JDK
Ce TrustStore est fourni en standard avec le JRE/JDK. Il contient les certificats de nombreuses autorités de certification racines.

Sommaire

Afficher le contenu d'un magasin de clés

```
$ keytool -list -storetype JCEKS -keystore TrustStore.jceks \
  -storepass ******** -v
```

Paramètres

- -storetype : spécifie le type de magasin de clés
- -keystore : spécifie le nom du fichier du magasin de clés
- -storepass : spécifie le mot de passe du magasin de clés pour vérifier l'intégrité
- -v : affiche le certificat sous la forme de caractères
- -rfc : affiche le certificat au format PEM (Base64)

Contrôle d'intégrite

Si l'option storepass n'est pas utilisée ou si le mot de passe n'est pas renseigné, voici le message d'avertissement : *l'intégrite des informations enregistrées dans votre Keystore n'a PAS ete verifiée ! Pour cela, vous devez spécifier le mot de passe de votre Keystore.*

Sommaire

Importer un certificat

```
$ keytool -importcert -storetype JCEKS -keystore TrustStore.jceks \
  -storepass ******** -alias CA -file ca.crt
```

Paramètres

- -storetype : spécifie le format du magasin de clés
- -keystore : spécifie le nom du fichier du magasin de clés
- -storepass : spécifie le mot de passe du magasin de clés pour garantir l'intégrité
- -alias : spécifie l'identifiant du certificat
- -file : spécifie le nom du fichier du certificat
- -trustcacerts : utilise le magasin de clés du JRE/JDK pour vérifier si le certificat est émis par une autorité de certification de confiance (si besoin)

Faire confiance à ce certificat ?
Cette question est posée à l'utilisateur si l'autorité de certification ayant signé le certificat n'est pas présente dans le magasin de clés (ou dans le magasin de clés du JRE/JDK).
Il est légitime de répondre "oui" uniquement pour l'ajout du certificat de l'autorité de certification racine.

Sommaire

Créer un certificat

Etape 1/3 : Générer la bi-clé

```
$ keytool -genkeypair -storetype JCEKS -keystore KeyStore.jceks \
  -keyalg RSA -keysize 2048 -storepass ******** \
  -keypass ******** -alias CertificatServer \
  -dname "CN=www.exemple.fr, O=Exemple SA, C=FR"
```

Paramètres

- -storetype : spécifie le format du magasin de clés
- -keystore : spécifie le nom du fichier du magasin de clés
- -keyalg : spécifie l'algorithme de génération de la bi-clé (RSA, DSA, EC, ...)
- -keysize : spécifie la taille de la bi-clé (à définir en fonction de l'algorithme)
- -storepass : spécifie le mot de passe du magasin de clés pour vérifier l'intégrité
- -keypass : spécifie le mot de passe pour garantir la confidentialité de la clé privée
- -alias : spécifie l'identifiant du certificat
- -dname : spécifie le "Distinguished Name" du certificat. Les éléments attendus sont CN (Common Name), OU (Organization Unit), O (Organization) et C (Country).

Créer un certificat

Etape 2/3 : créer la demande de certificat

```
$ keytool -certreq -sigalg SHA256withRSA -storetype JCEKS \
  -keystore KeyStore.jceks -alias CertificatServer \
  -file CertificatServer.csr
```

Paramètres

- -sigalg : spécifie l'algorithme de signature de la demande de certificat (SHA256withRSA, SHA1withDSA, SHA256withECDSA, ...)
- -storetype : spécifie le format du magasin de clés
- -keystore : spécifie le nom du fichier du magasin de clés
- -alias : spécifie l'identifiant du certificat
- -file : spécifie le nom du fichier de la demande de certificat

Obtenir le certificat

Envoyer la demande de certificat à l'autorité de certification pour obtenir un certificat signé par l'autorité de certification.

Créer un certificat

Etape 3/3 : importer le certificat signé

```
$ keytool -import -storetype JCEKS -keystore KeyStore.jceks \
  -storepass ******** -alias CertificatServer -file CertificatServer.crt
```

Paramètres

- -storetype : spécifie le format du magasin de clés
- -keystore : spécifie le nom du fichier du magasin de clés
- -alias : spécifie l'identifiant du certificat
- -storepass : spécifie le mot de passe du magasin de clés pour vérifier l'intégrité
- -file : spécifie le nom du fichier du certificat signé

Chaîne de certificat
Si ce n'est pas déjà fait, avant d'importer le certificat signé, il faut importer le certificat de l'autorité de certification racine et intermédiaire.

Sommaire

Créer une clé secrète

```
$ keytool -genseckey -storetype JCEKS -keystore KeyStore.jceks \
  -storepass ******** -keypass ******** \
  -alias MaCleSecrete -keyalg AES -keysize 256
```

Paramètres

- -storetype : spécifie le format du magasin de clés
- -keystore : spécifie le nom du fichier du magasin de clés
- -storepass : spécifie le mot de passe du magasin de clés pour vérifier l'intégrité
- -keypass : spécifie le mot de passe pour garantir la confidentialité de la clé secrète
- -alias : spécifie l'identifiant de la clé secrète
- -keyalg : spécifie l'algorithme de génération de la clé scrète (AES, DES, DESede, ...)
- -keysize : spécifie la taille de la bi-clé (à définir en fonction de l'algorithme)

Sommaire

Modifier l'alias d'un élément

```
$ keytool -changealias -storetype JCEKS -keystore KeyStore.jceks \
  -storepass ******** -alias MonCertificat -destalias MonAncienCertificat
```

Paramètres

- -storetype : spécifie le format du magasin de clés
- -keystore : spécifie le nom du fichier du magasin de clés
- -storepass : spécifie le mot de passe du magasin de clés pour vérifier l'intégrité
- -alias : spécifie l'identifiant de l'élément
- -destalias : spécifie le nouvel identifiant de l'élément


fin

Sources

Liens

- Manuel de keytool (Documentation Java 8)
- Java Cryptography Architecture Oracle Providers Documentation for JDK 8
- Classe "KeyProtector" du package "sun.security.provider" (CSP Sun)
- Classe "KeyProtector" du package "sun.crypto.provider" (CSP SunJCE)

Livres

- Pankaj Kuma. J2EE Security for Servlets, EJBs and Web Services : Applying Theory and Standards to Practice : Prentice Hall Professional, 2004