# Configuració certificats SSL des de Java 6/7 i JBoss 5.1/5.2

## Magatzems de claus per defecte

Quan es realitza una connexió SSL des de Java (per exemple per consumir un webservice que està sobre una adreça https), la implementació de Java comprova que el certificat de servidor està signat per una autoritat de certificació reconeguda.

Per fer-ho, empra un magatzem de claus («truststore»). Per cercar aquest magatzem, segueix les següents passes

- Si està definida la variable de sistema «javax.net.ssl.trustStore», empra el fitxer allà indicat. Cal indicar el password -a no ser que sigui una cadena buida-, amb la propietat « javax.net.ssl.trustStorePassword». Si el trustStore no és de tipus JKS, també cal indicar el tipus amb la propietat « javax.net.ssl.trustStoreType»
- Si no hi ha definida aquesta propietat cerca el fitxer %JAVA_HOME%\jre\lib\security\ jssecacerts, amb el password «changeit».
- Si aquest tampoc existeix, empra el fitxer per defecte, que es distribueix amb la màquina virtual: %JAVA_HOME%\jre\lib\security\cacerts, que té password per defecte «changeit».

Per tant, atès que per defecte no està fixada cap propietat, i el fitxer «jssecacerts» no existeix, el que sòl passar és que s'empra el fitxer «cacerts» que es distribueix amb la JVM.

Les actualitzacions de la màquina virtual poden afegir o llevar certificats al fitxer.

## Importar certificats al magatzem de claus

Quan connectam un servidor per «https» que té un certificat signat per una autoritat de certificació no reconeguda dins el «cacerts», o bé sigui un certificat autosignat per proves, en intentar connectar sortirà un error que normalment inclou la cadena:

> PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

Per solucionar-ho tenim tres opcions: incorporar el certificat del servidor a «cacerts», bé crear el fitxer «jssecacerts» amb el certificat, o bé crear un magatzem de claus amb un altre nom i referenciar-lo amb la propietat de sistema «javax.net.ssl.trustStore».

Qualcuns avantatges i inconvenients dels diferents sistemes:

- Si l'afegim a «cacerts», no cal fer res més, però directament aquests certificats s'aceptaran per totes les aplicacions que emprin aquesta màquina virtual, no només la nostra. D'altra banda, si actualitzam la màquina virtual, aquesta incorporarà un nou «cacerts» amb noves autoritats incorporades, i antigues llevades, i haurem de tornar incorporar el nostres

certificats. Per aplicacions en producció d'altra banda, potser no volem totes les autoritats de certificació que venen per defecte amb la màquina virtual.

- Si cream «jssecacerts», tampoc cal fer res més, però també directament aquests certificats s'acceptaran per totes les aplicacions que emprin aquesta màquina virtual. Si actualitzam la màquina virtual bastarà copiar el «jssecacerts» dins la nova. D'altra banda, el «jssecacerts» no inclou totes les autoritats de certificació ja preconfigurades del «cacerts».
- Si empram un «truststore» especificat amb la propietat «javax.net.ssl.trustStore» aquest servirà només per la nostra aplicació (o qualsevol que l'apunti amb la propietat), i no afectarà tampoc actualitzacions de la JVM. D'altra banda, com en el cas de «jssecacerts», no inclou totes les autoritats de certificació ja preconfigurades del «cacerts».

En qualsevol dels casos, per importar el certificat, o bé ens han de passar l'arxiu «pem» o bé el podem aconseguir. Amb les següents passes:

a) Accedint amb un navegador Firefox a la URL https://servidor/..., i anat a "Informació del lloc web" (pitjant damunt el pany), i "Connexió segura" (pitjar la fletxa cap a la dreta), i "Més informació", i a la nova finestra pitjar el botó "Mostra Certificat". A la pàgina que es carrega anar a l'enllaç "Baixa: PEM (cert)" i guardar l'arxiu "servidor.pem" (Altres navegadors ofereixen possibilitats similars)

b) En Linux (on consola Cygwin o WSL):

   - `openssl s_client -showcerts -connect servidor:443 </dev/null 2>/dev/null | openssl x509 -outform PEM > sevidor.pem`

## Importar certificat a «cacerts»

Executar el següent:

```
keytool -importcert -trustcacerts -file C:\servidor.pem -alias servidor -
keystore %JAVA_HOME%\jre\lib\security\cacerts -storepass changeit
```

## Importar certificat a «jssecacerts»

Executar el següent (si no existeix, el crearà):

```
keytool -importcert -trustcacerts -file C:\servidor.pem -alias servidor -
keystore %JAVA_HOME%\jre\lib\security\jssecacerts -storepass changeit
```

Important! No canviar el password, ha de ser "changeit"

## Importar a un truststore qualsevol

Executar el següent (si no existeix, el crearà):

```
keytool -importcert -trustcacerts -file C:\servidor.pem -alias servidor -
keystore myTrustStore.jks -storepass myPassword
```

Assegurar-se que el java que tenim dins el PATH coincideix amb el JAVA_HOME. El més senzill per estar segur:

```
set JAVA_HOME=C:\.....
set PATH=%JAVA_HOME%\bin;%PATH%
```

# Configurar JBoss 5.1/5.2

Si es fiquen els certificats dins «%JAVA_HOME%\jre\lib\security\cacerts» no cal fer res més.

Si es crea un truststore amb el nom «%JAVA_HOME%\jre\lib\security\jssecacerts» i password «changeit» tampoc no cal fer res més.

Si es crea un truststore amb un nom diferent, perquè les aplicacions que facin connexions l'emprin cal definit les propietats de sistema corresponents.

Per fer-ho es pot fixar dins el JAVA_OPTS als fitxers d'arrancada de JBoss. Si esà fixada la variable d'entorn JBOSS_HOME i es coloca el truststore dins aquest:

1. Windows, afegir al %JBOSS_HOME%\bin\run.conf.bat:
   - `set "JAVA_OPTS=%JAVA_OPTS% -Djavax.net.ssl.trustStore=%JBOSS_HOME%\ myTrustStore.jks -Djavax.net.ssl.trustStorePassword=myPassword"`
2. Linux, afegir al $JBOSS_HOME/bin/run.conf
   - `JAVA_OPTS="$JAVA_OPTS -Djavax.net.ssl.trustStore=$JBOSS_HOME/myTrustStore.jks -Djavax.net.ssl.trustStorePassword=myPassword"`

També és pot emprar qualsevol altre mecanisme per fixar system properties.

Per exemple, crear un fitxer que acabi amb el nom "-service.xml", per exemple, configuraciossl-service.xml, i copiar-lo dins un dels directoris de deploy, o directament copiar el fragment <mbean> a dins el fitxer JBOSS_HOME/server/default/conf/jboss-service.xml

```
      <server>
      ...
      <mbean code="org.jboss.varia.property.SystemPropertiesService"
name="jboss:type=Service,name=ConfiguracioSSLProperties">
            <attribute name="Properties">
                  javax.net.ssl.trustStore=C:\\jboss-eap-5.2.0\\myTrustStore.jks
                  javax.net.ssl.trustStorePassword=myPassword
            </attribute>
      </mbean>
      ...
      </server>
```

# Possibles errors

- PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
  - S'ha intentat connectar amb un servidor SSL que té un certificat que no està acceptat. Probablement fa falta importar el certificat dins el truststore que s'empri (ja sigui cacerts, jssecacerts o específic)
- java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty
  - S'ha especificat un truststore però no troba el fitxer. Verificar que la propietat "javax.net.ssl.trustStore" apunta al fitxer correcte. Verificar el JAVA_OPTS que

imprimeix el JBoss quan arranca, per exemple si s'empra la variable d'entorn JBOSS_HOME per fer referència al fitxer i aquesta no està definida.

# Alternatives

Alternativament, en funció de la llibreria client que s'empri, normalment solen tenir mecanismes per proporcionar programàticament un objecte SSLContext configurat, de manera que en aquest es pot sobreescriure el truststore, llegint manualment un fitxer truststore determinat, perquè només empri els certificats d'aquest, quan s'empri aquell client.

Es poden veure exemples de com realitzar aquesta configuració a la classe «FNMTCloudSignatureWebPlugin» mètode «prepareSsl» de portafib (que empra la llibreria httpclient de apache per fer les connexions SSL).

La llibreria CXF d'altra banda, també permet especificar a un fitxer de configuració dades del truststore a emprar per connexions SSL. Veure: http://cxf.apache.org/docs/client-http-transport-including-ssl-support.html#ClientHTTPTransport(includingSSLsupport)-ConfiguringSSLSupport

# Documentació de referència

- https://docs.oracle.com/javase/7/docs/technotes/guides/security/jsse/JSSERefGuide.html#CustomizingStores
- https://docs.oracle.com/javase/7/docs/technotes/tools/windows/keytool.html