Epayment Muc SSL Fehler

Fehler

Bei der Einrichtung des Tomcat auf bisto kamen es zu Problemen mit der SSL Verbindung. Der Handshacke ist Fehlgeschlagen.

Fehlermeldung

```
javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find
valid certification path to requested target.
```

Lösung

Nach der Anleitung von http://magicmonster.com/kb/prg/java/ssl/pkix_path_building_failed.html konnte das Problem gelöst werden.

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX p
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certi
```

We experienced this issue when a server changed their HTTPS SSL certificate, and our older version of Java did not recognize the root certificate authority (CA).

If you can access the HTTPS URL in your browser then it is possible to update Java to recognize the root CA.

In your browser, go to the HTTPS URL that Java could not access. Click on the HTTPS certificate chain (there is lock icon in the Internet Explorer, or the domain name left of the URL in firefox) and navigate the certificate hierarchy. At the top there should be a Primary Root CA. This could be missing from your java cacerts file. Note down the Issuer and Serial Number.

To verify the root certificates, determine where the cacerts file is located. By default it is in jre/lib/security/cacerts. The default password for this keystore is 'changeit'.

e.g. on my machine, I have both JDK and JRE, here is where they are located.

```
./jdk1.6.0_24/jre/lib/security/cacerts
./jre1.6.0_24/lib/security/cacerts
```

Different versions of java can have different cacerts.

If you do not want to modify the default JRE store, you can make a copy, and use the following system properties to specify the location and password.

```
javax.net.ssl.trustStore
javax.net.ssl.trustStorePassword
```

Once you have your keystore, dump its contents by using the list option.

```
keytool -list -v -keystore /path/to/cacerts  > java_cacerts.txt
Enter keystore password:  changeit
```

In this example, /path/to/cacerts is the location of your cacerts file, and the output of the command will be saved in java_cacerts.txt.

Take a look at java_cacerts.txt. See if it includes the same certificate that is present in the browser by searching for a matching serial number. In the java_cacerts.txt file, the serial number will be in lowercase and without the ":" colon character. If it is not present, then this could be the reason for the error, and we can fix this by adding the certificate found in the browser.

Back in the browser, export the Root CA. Choose the "X.509 Certificate (DER)" type, so the exported file has a der extension.

Assuming the file is called example.der, pick the alias 'example' for this certificate. Next import the file.

```
keytool -import -alias example -keystore  /path/to/cacerts -file example.der
```

You will be prompted for a password, use 'changeit'

and respond "yes" on whether to trust this key.

Dump the contents again to verify it contains your new certificate. Restart the JVM and check that it can now access the HTTPS URL. Also remove the java_cacerts.txt dump file.

See also java-samples.com and keytool.

Published: Thursday, 31 March 2011