The Race Condition of Conflicting SSL Certificate Stores

Posted At : July 22, 2004 1:02 PM | Posted By : Steven Erat
Related Categories: Java, Learning, ColdFusion

According the behavior I've observed today, a JVM loads one and only one SSL certificate store (keystore) at runtime. The JVM under ColdFusion comes with a default keystore having many popular SSL certificate types already imported into it. That default keystore is {cfmx}/runtime/jre/lib/security/cacerts. However, if you programmatically load a different keystore, then you will encounter a race condition where only one of the two certificate stores is loaded. The certificate store that gets loaded first is used by the JVM for all subsequent SSL connections. If the certificate stores have disparate certificates imported into them, then some SSL connections probably won't work and will vary back and forth every time the server restarts.

(Note that in the example test described here, there were 2 SSL websites used, and neither of which had a certificate already in a keystore. One application used CFHTTP to connect to one SSL site, and another application used a CFX Java to connect to a different SSL site. The certificate used for the CFHTTP SSL connection was imported into the default keystore, and the certificate used for the CFX SSL connection was imported into the alternate keystore.)

In this specific case, I was using a CFX Java custom tag that loaded its own certificate store and made its own SSL HTTP connection. The Java source in the CFX tag that loads the alternate keystore is
System.setProperty("javax.net.ssl.trustStore",truststorelocation)
where **truststorelocation** might be something like **C:Tempalt_cacerts** to indicate a different certificate store. I confirmed that the CFX tag worked as expected.

But also in this example, if I *first* ran a CFHTTP to an SSL enabled website just after the ColdFusion server started, then the ColdFusion server would load the default **jre/lib/security/cacerts** keystore instead which had the appropriate certificate for that SSL site. The CFHTTP tag worked on the SSL site and returned the expected result. If I then ran the CFX tag after having run the CFHTTP test, the CFX tag would attempt to load its own keystore to connect to a different SSL site that required a different certificate. When run *after* the CFHTTP tag, the CFX would always fail with the error:

sun.security.validator.ValidatorException: No trusted certificate found

To test the opposite scenario, I restarted CFMX then immediately requested the CFX Java tag, again using its own defined location of a certificate store. The CFX worked. Running the CFHTTP tag to a different SSL site immediately after the CFX tag, the CFHTTP tag would fail with the error message:

I/O Exception: peer not authenticated

So the default keystore under the JVM had imported certificate appropriate for the SSL website used in the CFHTTP connection, and the alternate keystore used in the CFX Java tag had a different set of certificates imported into it which were appropriate for the other SSL website. After ColdFusion restarted, which ever application loaded first, the CFX or the CFHTTP, would determine which certificate store would be loaded, and one of the two applications would fail.

It seems that the best solution for this would be to consistently use just one keystore for all the server's SSL needs. Import all the necessary certificates into the default certificate store, then let the ColdFusion server use only that one keystore.