Tomcat中证书库文件口令加密实现指导

**场景描述：**

利用Tomcat配置Https连接时，需要在配置文件server.xml中对433端口做如下配置：

|  |
| --- |
| <Connector server='tcc' port="18443" protocol="HTTP/1.1" SSLEnabled="true"  maxThreads="150" scheme="https" secure="true"  clientAuth="false" sslProtocol="TLS" keystoreFile="/opt/huawei/TCCProd/apache-tomcat-7.0.78/webapps/TaskControlCenter/WEB-INF/classes/cacerts/tcc.keystore" keystorePass="Huawei123"/> |

其中，protocol表示连接器的类型，Tomcat有默认连接器、Nio连接器、ARP连接器三种连接器可选。KeystoreFile表示证书库文件的地址，keystorePass表示证书库文件的口令。然而这里配置的口令只能明文显示，这不符合网络安全红线的规定。

**方案描述：**

为了解决Tomcat配置文件中口令明文显示的问题，就要从server.xml中先于Tomcat的连接器去截取事先放置好的密文口令并进行解密处理，这样Tomcat的连接器处理的时候已经转化为可以处理的明文口令了。基本的思路就是：创建一个新的连接器，并且继承Tomcat自身的连接器，先将server.xml中所事先配置的keystorePass进行解密处理，再初始化父类连接器。以Tomcat默认的连接器为例，代码样例如下：

|  |
| --- |
| **public** **class** TCCHttp11Protocol **extends** Http11Protocol {    @Override  **public** **void** init() **throws** Exception {  String password = getKeystorePass();  String realpassword = decipher(password.trim());  setKeystorePass(realpassword);  **super**.init();  }  **private** String decipher(**final** String password) {  // 这里执行密码的解码操作;  String keyfilePath = TCCHttp11Protocol.**class**.getClassLoader().getResource("").getPath();    **byte**[] workKey = CryptUtil.*genWorkKey*(keyfilePath+TccCrypt.*Key1*, keyfilePath+TccCrypt.*Key2*, keyfilePath+TccCrypt.*Key3*);  **byte**[] salt = CryptUtil.*genSalt*(keyfilePath+TccCrypt.*Salt*);  **return** CryptUtil.*decryptData*(password, workKey, salt);  }  } |

由此思路编写插件并编译打包为jar包。

加密操作流程如下：

# 使用插件对口令keystorepass进行加密



此文件是编译好的Tomcat插件，以Tomcat7为例，这个插件可以让Tomcat先解密口令，再初始化默认的连接器。

将此文件上传到Tomcat的lig目录，并解压 unzip TCCTomcatPlugin-1.0.zip

加密证书库文件口令为密文，以将口令“keystorepass”用AES/GCM/NoPadding加密，在Tomcat7的lib目录下，输入以下命令：

|  |
| --- |
| /opt/huawei/TCCProd/apache-tomcat-7.0.78/lib> java -cp .:./\* com.huawei.platform.tcc.TccCrypt keystorepass  encrypted keystorePass:  32XvxESEnp73EBgLuVl0K6RC76d6C39xsFL0PA== |

程序将输出证书库文件口令“keystorepass”使用AES/GCM/NoPadding加密后的结果为：“32XvxESEnp73EBgLuVl0K6RC76d6C39xsFL0PA==”。

在此步骤中，有可能会报出错误：“Can not initialize cryptographic mechanism”。

|  |
| --- |
| Exception in thread "main" java.lang.ExceptionInInitializerError  at javax.crypto.Cipher.getInstance(Cipher.java:643)  at javax.crypto.Cipher.getInstance(Cipher.java:585)  at com.huawei.platform.tcc.CryptUtil.encryptToAESStr(CryptUtil.java:64)  at com.huawei.platform.tcc.CryptUtil.encryptData(CryptUtil.java:343)  at com.huawei.platform.tcc.TccCrypt.main(TccCrypt.java:33)  Caused by: java.lang.SecurityException: Can not initialize cryptographic mechanism  at javax.crypto.JceSecurity.<clinit>(JceSecurity.java:86)  ... 5 more  Caused by: java.lang.SecurityException: Cannot locate policy or framework files!  at javax.crypto.JceSecurity.setupJurisdictionPolicies(JceSecurity.java:256)  at javax.crypto.JceSecurity.access$000(JceSecurity.java:48)  at javax.crypto.JceSecurity$1.run(JceSecurity.java:78)  at java.security.AccessController.doPrivileged(Native Method)  at javax.crypto.JceSecurity.<clinit>(JceSecurity.java:76)  ... 5 more |

这是因为这个是jre中使用了默认的安全包。但是这包支持的密码长度范围短，无法满足北美标准。用下面的jar包替换目录“$JAVA\_HOME/jre/lib/security/ ”中的jar文件（一般路径为/usr/java/latest/jre/lib/security/）：



替换jre的默认安全包，可以使其满足北美标准，支持更长的密码。

# 在connector中使用加密后的keystorepass

更新Tomcat的配置文件server.xml，修改protocal的值为com.huawei.platform.tcc.TCCHttp11Protocol，修改keystorePass的值为步骤1中加密后的值：

|  |
| --- |
| <Connector server='tcc' port="18443" protocol="com.huawei.platform.tcc.TCCHttp11Protocol" SSLEnabled="true"  maxThreads="150" scheme="https" secure="true"  clientAuth="false" sslProtocol="TLS" keystoreFile="/opt/huawei/TCCProd/apache-tomcat-7.0.78/webapps/TaskControlCenter/WEB-INF/classes/cacerts/tcc.keystore" keystorePass="32XvxESEnp73EBgLuVl0K6RC76d6C39xsFL0PA=="/> |

其中protocol="com.huawei.platform.tcc.TCCHttp11Protocol "，表示使用定制的连接器来代替默认的连接器。keystorePass="32XvxESEnp73EBgLuVl0K6RC76d6C39xsFL0PA=="，表示证书库文件的口令改为密文。

重启Tomcat后就设置成功了。