使用CA签发的服务器证书搭建TOMCAT双向SSL认证服务

Published by 翊天 on 2011年4月28日 | Leave a response

作者：junsan
QQ：334620162
发布网址：http://www.javastar.org/?p=120

这周一个项目上线，使用了SSL双向验证的方式保护WebService接口，原本自己使用keytool签发的服务器和客户端证书在开发和测试的时候都是正常的，但是，在上线后，通过公司的CA平台签发出来的正式证书，却始终无法通过java客户端正常链接。

后来通过网上的一个InstallCert.java的文件生成本地可信任证书库，倒是可以连接了，但是总觉得不对劲，后来仔细检查了一下，原来自己的操作步骤中漏了一步。

为了备忘，也防止自己以后再犯类似的错误，记录下来以备参考，同时也提供给需要的朋友，省的走弯路。

本文使用java的jdk自带的keytool为例说明。

第一部分，先说证书的申请。

这步是要到正规的CA公司申请正式的设备证书必须走的步骤。

1、先生成证书的密钥对

打开命令行，切换到某个自己新建的目录下，执行如下命令

keytool -genkey -keyalg RSA -keysize 1024 -dname “CN=www.javastar.org,OU=翊天阁,O=翊天阁,L=南京市,ST=江苏省,C=CN” -alias server -keypass 123456 -keystore server.jks -storepass 123456 -validity 365

这里说明一下几个重要的地方,CN=www.javastar.org，这里的www.javastar.org一定要换成你实际要部署的站点的域名，如果是在内网，就要用服务器的hostname，一定不可以用IP，否则，是无法建立SSL链接的。

OU=翊天阁,O=翊天阁，这里可以替换成你自己的组织名称，或者公司名称。

-validity 365这里声明证书有效期为1年。

其他的参数自己可以参考keytool的使用帮助或相关文档。

好了，成功执行上面的命令后，我们在当前命令行所在目录得到server.jks文件，这个就是包含密钥对的基本证书信息库文件。

2、导出证书请求文件

keytool -certreq -alias server -sigalg “SHA1withRSA” -file server.pem -keypass 123456 -keystore server.jks -storepass 123456

这里可以得到一个server.pem的文件。

3、向CA公司申请签发设备证书

将上一步得到的server.pem证书发送给相关的CA公司，CA公司会通过这个申请签发一张设备证书，最后我们会得到一个.cer的文件，比如server.cer。

同时，我们要取得该CA公司的证书链，比如会有CA_ROOT.cer和CA_CA.cer，第一张为CA公司的根证书，第二张为CA公司的签名证书。

4、将CA根证书导入服务器证书库

keytool -import -alias CA_ROOT -keystore server.jks -trustcacerts -storepass 123456 -file CA_ROOT.cer

5、将CA签名证书导入服务器证书库

keytool -import -alias CA_CA -keystore server.jks -trustcacerts -storepass 123456 -file CA_CA.cer

6、使用CA签发的证书回复我们自己生成的包含私钥的证书

keytool -import -alias server -keystore server.jks -trustcacerts -storepass 123456 -file server.cer

7、导出回复成功后的服务器证书

keytool -export -alias server -storepass 123456 -file javastar.org.cer -keystore server.jks

好了，到这里我们需要的设备证书已经ok了。

最后对我们实际有用的是两个文件：server.jks，是服务器证书库，存储了含有私钥的服务器证书，及其证书链，这个就是主要的设备证书了，是放在服务器的SSL配置里面；还有一个是javastar.org.cer，这个是只包含服务器证书公钥的设备证书，是发给用户，让用户放入自己的可信任库的。

第二部分，配置**Tomcat**的**SSL**双向链接

1、准备客户端证书

keytool -genkey -v -alias client -keyalg RSA -storetype PKCS12 -keystore client.p12 -dname “CN=Client,OU=javastar.org,L=nj,ST=js,C=cn” -storepass 123456 -keypass 123456 -validity 365

执行完毕，我们会得到一张p12的客户证书client.p12。

2、导出.cer格式的客户证书

keytool -export -alias client -keystore client.p12 -storetype PKCS12 -storepass 123456 -rfc -file client.cer

执行完毕得到client.cer证书。

3、将客户端证书导入服务器的可信任证书库

keytool -import -v -file client.cer -keystore servertrust.keystore -storepass 123456

执行完毕得到servertrust.keystore，这个是给tomcat服务器端用的。

4、将服务器证书导入客户端可信任证书库

keytool -import -v -file javastar.org.cer -keystore clienttrust.keystore -storepass 123456

执行完毕会得到clienttrust.keystore证书库，留着备用。同时，也可以同样的方法，把CA_ROOT.cer和CA_CA.cer导入。

5、在和tomcat的bin目录并行的目录下新建一个cert文件夹，把server.jks和servertrust.keystore文件拷贝进去。

6、找到tomcat的server.xml文件，找到如下的内容

```
<!–
  <Connector port=”8443″ protocol=”HTTP/1.1″ SSLEnabled=”true”
        maxThreads=”150″ scheme=”https” secure=”true”
        clientAuth=”false” sslProtocol=”TLS” />
  –>
```

去掉注释，并修改为

```
  <Connector port=”8443″ protocol=”HTTP/1.1″ SSLEnabled=”true”
        maxThreads=”500″ scheme=”https” secure=”true”
        clientAuth=”true” sslProtocol=”TLS”
        keystoreFile=”D:/apache-tomcat-6.0.20/cert/server.jks”
keystorePass=”123456″
     truststoreFile=”D:/apache-tomcat-6.0.20/cert/servertrust.keystore”
truststorePass=”123456″/>
```

这里注意把D:/apache-tomcat-6.0.20/cert修改为你的tomcat实际的路径。

7、启动或者重启tomcat，在浏览器中导入client.p12的证书，然后，在地址栏输入https://www.javastar.org:8443，就会弹出证书选择框，选择证书后，就可以使用SSL协议访问tomcat服务器了。

第三部分，在程序中如何建立与服务器的**SSL**双向认证链接

其实这个就简单了，以java为例，比如我们现在要访问一个通过SSL双向认证保护的WebService接口，只需要在生成的客户端程序中开始部分加入

```
System.setProperty(“javax.net.ssl.keyStore”, “D:/client.p12″);
System.setProperty(“javax.net.ssl.keyStorePassword”, “123456″);
System.setProperty(“javax.net.ssl.keyStoreType”, “PKCS12″);
System.setProperty(“javax.net.ssl.trustStore”, “D:/clienttrust.keystore”);
System.setProperty(“javax.net.ssl.trustStorePassword”, “123456″);
System.setProperty(“javax.net.ssl.trustStoreType”, “JKS”);
```

JVM会自动将证书提交给服务器验证，由于服务器证书也在我们的可信任库，也会自动信任服务器端的证书。

下面的代码可以在需要的时候清除以上的环境内容

System.clearProperty(“javax.net.ssl.keyStore”);
System.clearProperty(“javax.net.ssl.keyStorePassword”);
System.clearProperty(“javax.net.ssl.keyStoreType”);
System.clearProperty(“javax.net.ssl.trustStore”);
System.clearProperty(“javax.net.ssl.trustStorePassword”);
System.clearProperty(“javax.net.ssl.trustStoreType”);

有一点要提醒的就是，客户端连接的时候，需要把连接的url修改为https的，如以JAX-WS的客户端为例

URL url = new URL(“https://www.javastar.org:8443/WsPort.ws?wsdl”);//这里就是实际发布的wsdl的地址
WsService hws = new WsService(url);//自动生成的类可能没有这个方法，你需要仿照写一个
Ws ws = hws.getUsvsPort();
自建的构造方法：
public WsService(URL wsdlLocation) {
 super(wsdlLocation, new QName(“http://webservice.javastar.org/“,
  ”WsService”));
}

得到了对象的实例，其他的就跟我们普通操作java类一样了。

如果发现问题可以和我交流，QQ334620162，欢迎转载，请保留版权信息。

相关文章

- 软件架构和框架研究之二：软件框架的概念、与架构的关系及分类
- 软件架构和框架研究之一：软件架构的概念来源及分类
- java原理:java的内省机制
- java原理:java的反射机制
- sping 3.2.x 源代码分析之八:Spring的各类BeanDefinitionParser解析器加载和使用
- sping 3.2.x 源代码分析之七:DispatcherServlet加载spring context的过程
- sping 3.2.x 源代码分析之六:ContextLoaderListener加载spring context的过程
- sping 3.2.x 源代码分析之五:创建spring的测试工程

Posted in JAVA开发, WEB技术
Tagged ca, ssl, tomcat, 证书

LEAVE A REPLY

Name

Email

Website

Comment

POST COMMENT

← PREVIOUS

NEXT →

文章分类

- CMS研究
- C和C++

- JAVA开发
- WEB技术
- 业界新闻
- 云计算
- 人生历程
- 企业系统
- 信息安全
- 创业之路
- 协议原理
- 团队建设
- 操作系统
- 数据库技术
- 移动开发
- 计算机应用
- 软件工程

标签

aop apache c++ centos cookie crm eclipse google hibernate hostname http IE IP java javascript jboss jquery linux maven myeclipse mysql oracle php policy postgresql redhat security session spring struts tomcat ubuntu windows wordpress 人生 工作 快捷键 数据库 架构 架构设计 源代码分析 英语 设计 证书 软件架构

功能

- 登录
- 文章RSS
- 评论RSS
- WordPress.org

秋装男士休闲裤修身型

1
2
3
4
5
6
7
8
9
10

友情链接：启天网 启天论坛 启天合租 春风化羽设计网