

Apache Tomcat Servlet/JSP 容器如何设置 SSL

测试环境:

- Windows 2003
- Tomcat 6.0.18

文档说明: 本文件适用于用 JSSE 设置 Tomcat 的 SSL。如果使用 APR, Tomcat 是通过 OpenSSL 来实现 SSL, 设置会不同。

SSL 和 Tomcat

必须要注意, 配置 Tomcat 来利用 secure sockets 仅仅在它作为一个独立的 web 服务器时才有必要。当 Tomcat 主要作为在另外一个 web 服务器, 如 Apache 或 Microsoft IIS, 后面的 Servlet/JSP 容器运行时, 通常有必要把主要的 web 服务器配置来处理与用户的 SSL 连接。通常, 这个服务器会对所有的 SSL-相关的功能进行交涉, 然后再把对 Tomcat 容器的请求解密后传递过去。同样, Tomcat 会返回明码的回应, 这个回应将被加密后才被送到用户浏览器。在这样的环境下, Tomcat 知道与主要 web 服务器和客户的交流是通过一个安全连接才发生的(因为你的程序需要询问这些情况), 但是它本身并没有参与加密和解密。

使用 SSL 需要考虑的地方

用户第一次试图到你的网站访问被安全保卫的页面时, 他(她)通常会被要求提供关于认证的详细信息(如公司名称和联系姓名), 并询问他(她)是否愿意接受这个认证的合法性, 并继续进行交易。一些浏览器会提供一个选项来永久地接受所给的认证的合法性, 这样一来用户就不用每次访问你的网站时麻烦地去填写认证信息。有的浏览器没有这一选项。一旦用户认可后, 这个认证至少在浏览器会话期间被认为是合法的。

虽然 SSL 协议的设计尽量做到既安全又高效, 但是加密和解密是复杂耗时的计算过程。通常没有必要让整个网站都通过 SSL 来传输, 开发人员可以选择部分网页用 SSL, 部分网页不用 SSL。对于一个相对繁忙的网站来说, 可以选择保护那些比较敏感的信息, 如登陆、个人信息、购物车、付账、信用卡等。那些网页只要把网页地址里 http: 改为 https:, 浏览器就会自动把信息按照指定的协议传输出去。

最后, 使用根据网名决定的虚拟主机来进行安全连接可能会出现问题。这是 SSL 协定本身的设计局限。SSL handshake, 就是客户浏览器接受服务器认证, 在 HTTP 请求访问之前必须产生。

地址: 上海市徐汇区桂林路 396 号西南软件园 3 号楼 (原 29 号楼) 611 室 (200233)

统一服务电话: 400 600 9103

电话: 8621-64701090

传真: 8621-54640133-803

E-mail: caich8@hotmail.com

网址: <http://www.wang-cai.com>

因此，包含虚拟主机名字的请求信息在认证之前不能被确定，所以不可能给单个 IP 地址分配多个证书。如果在单个 IP 地址上的所有的虚拟主机需要对照同一个证书来认证的话，再添加多个虚拟主机不应该干扰服务器上正常的 SSL 操作。不过要当心，大多数的客户浏览器会按证书上列出的 domain names (主要是官方的，CA-签署的证书) 对照比较服务器的 domain name。如果领域名(domain names)不相配，这些浏览器会向客户端用户显示警告。总的来说，只有 address-based 虚拟主机通常和 SSL 一起在生产环境中被使用。

设置

产生 Keystore

Tomcat 现在只支持 JKS 或 PKCS12 格式的 keystores. JKS 格式是 Java 的标准 KeyStore 格式，它可以用 Java 的 keytool 来产生。这个工具在 Java 的 bin 目录里。PKCS12 格式是互联网的标准，可以用 OpenSSL 和微软的 Key-Manager 来修改。

要从头开始产生一个新的 keystore，包含一个自签的证书，从一个终端命令行执行下面的命令：

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA
```

(应该把 RSA 运算法则作为主要安全运算法则，这保证了与其它服务器和组件的兼容性。)

这个命令会在用户的 home directory 产生一个叫做 ".keystore" 的新文件。要指定一个不同的位置(location)或文件名，在上面所示的 keytool 命令里添加 -keystore 参数，后面紧跟着你的 keystore 文件的全部路径名。你还需要把这个新的位置在 server.xml 配置文件中反映出来，这在后面将有描述。例如：

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA \
-keystore \path\to\my\keystore
```

在执行这个命令后，你首先被要求出示 keystore 密码。Tomcat 使用的默认密码是 "changeit" (全都是小写字母)，如果你愿意，你可以指定你自己的密码。你还需要在 server.xml 配置文件里指定自己的密码，这在以后会有描述。

下一步，你会被要求出示关于这个证书的一般性信息，如公司，联系人名称，等等。这些

地址：上海市徐汇区桂林路 396 号西南软件园 3 号楼（原 29 号楼）611 室 （200233）

统一服务电话：400 600 9103

电话：8621-64701090

传真：8621-54640133-803

E-mail: caich8@hotmail.com

网址: http://www.wang-cai.com

信息会显示给那些试图访问你程序里安全网页的用户，以确保这里提供的信息与他们期望的相对应。提示用户名时请输入网站域名。

最后，你会被要求出示 *密钥(key) 密码*，也就是这个认证书所特有的密码(与其它的储存在同一个 keystore 文件里的认证书不同)。你**必须**在这里使用与 keystore 密码相同的密码。(目前，keytool 会提示你按 ENTER 键会自动帮你做这些)。

如果一切顺利，你现在就拥有了一个可以被你的服务器使用的有认证书的 keystore 文件。

注意： 你的 private key 的密码和 keystore 的密码应该相同。如果不同的话你会得到一下错误信息： java.io.IOException: Cannot recover key 这是一个已知的错误，详细请看：[Bugzilla issue 38217](#)

Edit the Tomcat Configuration File

最后的步骤是把你的 secure socket 配置在 \$CATALINA_HOME/conf/server.xml 文件里，\$CATALINA_HOME 代表你在其中安装 Tomcat 5 的目录。一个例子是 SSL 连接器的 <Connector> 元素被包括在和 Tomcat 一起安装的缺省 server.xml 文件里。它看起来象是这样：

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<!--
<Connector
    port="8443" minProcessors="5" maxProcessors="75"
    enableLookups="true" disableUploadTimeout="true"
    acceptCount="100" debug="0" scheme="https" secure="true";
    clientAuth="false" sslProtocol="TLS"/>
-->
```

你会注意到 Connector 元素本身，其默认形式是被注释掉的(commented out)，所以你需要把它周围的注释标志删除掉。然后，你可以根据需要客户化(自己设置)特定的属性。关于各种选项的详细信息，请查阅 [Server Configuration Reference](#) 。下面的讨论仅仅涵盖设置 SSL 通信(communication)时大家最感兴趣的那些属性。

这个 port 属性(默认值是 8443)是 TCP/IP 端口数码，Tomcat 在其上监听安全连接。你可以把它更改成任何你愿意要的数值(如默认的 https 通信，数目是 443)。不过，在许多操作系统中，要想在比 1024 小的端口数码上运行 Tomcat，需要特殊的设置(它超出了这个文档资料的范围)。

如果你在这里更改端口数值，你还必须更改在 non-SSL 连接器上的 redirectPort 这个属性特定的值。这允许 Tomcat 自动地 redirect 那些试图访问有安全限制页面的用户，指明根据

地址：上海市徐汇区桂林路 396 号西南软件园 3 号楼（原 29 号楼）611 室 （200233）

统一服务电话：400 600 9103

电话：8621-64701090

传真：8621-54640133-803

E-mail: caich8@hotmail.com

网址: http://www.wang-cai.com

Servlet 2.4 Specification 要求，SSL 是必需的。

有一些额外的选项被用来配置 SSL 协定。依赖于你早先怎样配置你的 keystore，你也许需要添加或更改下列属性值：

属性	描述
clientAuth	如果你想要 Tomcat 要求所有的 SSL 客户在使用这个 socket 时出示用户认证书，把这个值设定为 true 。如果你想要 Tomcat 要求出示用户认证书，但是如果没有认证书也可以， 就把这个值设定为 want 。
keystoreFile	如果你产生的 keystore 文件不在 Tomcat 期望的默认地方(一个叫做 .keystore 的文件在 Tomcat 运行的主目录)，就添加这个属性。你可以指定一个绝对路径名称， 或者一个由 \$CATALINA_BASE 环境变量而派生的相对路径名称。
keystorePass	如果你使用一个不同的 keystore (以及认证书) 密码，而不是 Tomcat 期望的密码 (就是 changeit)，添加这个元素。
keystoreType	如果使用一个 PKCS12 keystore 的话，就添加这个 element。 有效的值是 JKS 和 PKCS12 。
sslProtocol	要在这个 socket 上被使用的加密 / 解密协定。如果你在使用 Sun 的 JVM，我们不提倡更改 这个值。据报道，TLS 协定的 IBM' s 1.4.1 实现与一些通用的浏览器不兼容。 如果是这样，就使用 value SSL 。
ciphers	这个 socket 允许使用的由逗号分隔开的加密密码列单。默认的情况下，任何可用的密码都允许被使用。
algorithm	可用的 x509 算法。默认是 Sun 的实现 (SunX509)。 对于 IBM JVMs，你应该使用值 IbmX509。对于其他卖主，查阅 JVM 文档资料来 找正确的值。
truststoreFile	用来验证用户认证书的 TrustStore 文件。
truststorePass	访问 TrustStore 的密码。默认值就是 keystorePass 的值。
truststoreType	如果你在使用与 KeyStore 不同格式的 TrustStore，添加这个元素。 合法的值是 JKS 和 PKCS12。
keyAlias	如果 keystore 里面有多 key，你可以为用这个选项为加入的 key 起一个名字。 如果没有指定名字，使用时 keystore 内的第一个 key 将会被使用。

在完成这些配置更改后，你必须象通常那样重新启动 Tomcat，然后你就可以工作了。你应该可以通过 SSL 访问 Tomcat 支持的任何 web 应用程序。例如，试一下下面的指令：

```
https://localhost:8443
```

你应该看到通常的 Tomcat splash 页面(除非你修改过 ROOT web 应用程序)。如果不行的话, 下面的章节包含一些排除故障的提示。

安装和获取认证权威颁发的证书

要从认证权威(如 verisign.com, thawte.com 或 trustcenter.de)获得并安装证书, 你应该先阅读前面章节, 然后再按照下面的指示进行:

产生一个认证签名请求 (CSR)

为了从你选择的认证权威那里获得证书, 你必须产生一个所谓的认证签名请求 (Certificate Signing Request (CSR))。这个认证签名请求被认证权威用来产生鉴定你的网站是“安全的”的证书。按照下列步骤产生一个 CSR:

- 产生一个局部证书(如前面章节描述那样):

```
keytool -genkey -alias tomcat -keyalg RSA \
    -keystore <your_keystore_filename>
```

- 注意: 在某些情况下, 要产生一个工作证书, 你必须在 “first- and lastname” field 键入你的网站的域名 (例如 www.jaxmao.org)。
- 然后用下列指令产生 CSR :

```
keytool -certreq -keyalg RSA -alias tomcat
    -file certreq.csr \
    -keystore <your_keystore_filename>
```

执行这个命令之后你会得到一个文件叫 certreq.csr。然后你可以到发送到认证权威, 然后你就可以得到你的证书。具体步骤可以参考他们的网站的帮助文件。

导入证书

现在你有了证书, 你可以把它输入到你局部的 keystore 里。首先你必须输入一个叫做证书链 Chain Certificate 或 Root Certificate 到你的 keystore 里去。在这之后, 你可以开始输入你的证书。

- 从你获得认证书的认证权威那里下载一个 Chain Certificate 。
- 把 Chain Certificate 输入到你的 keystore 里边

```
keytool -import -alias root -keystore <your_keystore_filename> \
-trustcacerts -file <filename_of_the_chain_certificate>
```

- 最后输入你的新的认证书

```
keytool -import -alias tomcat -keystore <your_keystore_filename> \
-file <your_certificate_filename>
```

- 在 CA Trust Store 中导入根证书: `keytool -import -keystore %JAVA_HOME%\jre\lib\security\cacerts -storepass changeit -file ca.cer -alias localhost`
- 在 server.xml 中加入 CA Trust Store 信息并且设置 `clientAuth="true"`
- 在浏览器中如果没有有效证书, 访问网站时显示网站错误; 如果有一个有效证书, 自动进入网站; 如果有多个有效证书, 会弹出窗口让用户进行选择。

Troubleshooting

可以在服务器上启动 SSL 相关日志:

```
set CATALINA_OPTS=-Djavax.net.debug=ssl
```

```
bin\startup
```

这里是一些你在设置 SSL 通信时也许会遇到的常见问题, 以及如何解决它们。

- 我在我的日志文件中得到“java.security.NoSuchAlgorithmException”错误。
JVM 找不到 JSSE JAR 文件。按照所有的指导说明 [下载并安装 JSSE](#)。
- 当 Tomcat 启动时, 我得到一个象 “java.io.FileNotFoundException: {some-directory}/{some-file} not found” 的异常。

一个可能的解释是 Tomcat 不能在它要找的地方找到 keystore 文件。在默认的

地址: 上海市徐汇区桂林路 396 号西南软件园 3 号楼 (原 29 号楼) 611 室 (200233)

统一服务电话: 400 600 9103

电话: 8621-64701090

传真: 8621-54640133-803

E-mail: caich8@hotmail.com

网址: <http://www.wang-cai.com>

情况下，Tomcat 预计 keystore 文件在 Tomcat 运行的用户主目录里被命名为 .keystore。如果 keystore 存放在别的什么地方，你需要向 Tomcat 配置文件里的 `<Factory>` 元素添加一个 `keystoreFile` 属性。

- 当 Tomcat 启动时，我得到一个象“`java.io.FileNotFoundException: Keystore was tampered with, or password was incorrect`”的异常。

如果没有其他人 试图 修改你的 keystore 文件，最可能的情况是 Tomcat 用了错误的密码。你可以尝试重新产生 keystore 文件，或修改 Tomcat 设置文件里 `<Connector>` 元素的 `keystorePass` 属性，还请记住密码是分大小写的。

如果你仍然有问题，一个好的地方你可找到帮助是 **TOMCAT-USER** 邮件列表。你可以用下面的地址订阅。 <http://tomcat.apache.org/lists.html>.

Miscellaneous Tips and Bits

你可以用下面的办法获取用户 SSL session 的 ID:

```
String sslID = (String)request.getAttribute("javax.servlet.request.ssl_session");
```

3.2 Deny revoked certificates (via the CRL)

For CRL to work in JBoss/Tomcat the

`${jboss.server.home.dir}/deploy/jbossweb-tomcat55.sar/tomcat-util.jar` must contain the classes:

- `org.apache.tomcat.util.net.jsse.JSSE15Factory`
- `org.apache.tomcat.util.net.jsse.JSSE15SocketFactory`

To arrange this it is necessary to do some recompilation with a JDK1.5 target. If you do not update this jar CRL will **NOT** work! A copy of the recompiled jar containing the required libraries is [here](#) on an as is basis, (I have no idea if this affects other functions, security concerns, performance etc... so use at own risk, don't run with scissors blah blah).

Create a CRL (see earlier on this page) and copy the CRL into the conf directory so that the server can access it.

```
cp crlFile.pem ${jboss.server.home.dir}/conf/server.crlFile
```

Edit `jbossweb-tomcat50.sar/server.xml` and include the extra line:

```
maxThreads="100" strategy="ms"
maxHttpHeaderSize="8192" emptySessionPath="true"
scheme="https" secure="true" clientAuth="true"
sslProtocol = "TLS"
keystoreFile="${jboss.server.home.dir}/conf/server.keystore"
keystorePass="123456"
truststoreFile="${jboss.server.home.dir}/conf/server.truststore"
```



```
truststorePass="123456"
```

```
crlFile="${jboss.server.home.dir}/conf/server.crlFile" />
```

Restart your JBoss server and attempt to reach it at for example:

<https://localhost:8443/jmx-console/index.jsp>

- This should still work.
- Now remove the good client certificate from your browser and install a revoked client PKCS12 certificate into your browser.
- Attempt to connect again. It should now fail. It may show a nice error it may not depending on the browser.
- Now remove the revoked one and install a good client PKCS12 certificate into your browser .
- Attempt to connect again. It should work again.

NOTES:

Some articles imply that the format for the crlFile is independent of the keystore format. **This does not appear to be the case.**

Broken Example: Here you have the keystore type set as PKCS12 even though the CRL is a PEM format file (and hence not PKCS12 format). PKCS12 contains the private key as well as the certificate and public key - which is not applicable for the CRL file so this seems to be why it's getting confused.

```
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" disableUploadTimeout="true"
acceptCount="100" scheme="https" secure="true"
clientAuth="true" sslProtocol="TLS"
crlFile="/ca/crl/crl.pem"
keystoreType="PKCS12"
keystoreFile="/ca/ssl/idp.p12"
keystorePass="123456" />
```

地址：上海市徐汇区桂林路 396 号西南软件园 3 号楼（原 29 号楼）611 室 （200233）

统一服务电话：400 600 9103

电话：8621-64701090

传真：8621-54640133-803

E-mail: caich8@hotmail.com

网址: <http://www.wang-cai.com>

The only way I know of making it work is to remove the `keystoreType` field and building the keystores as described earlier on this page.