SSLWithFORMFallback

This page describes a Tomcat setup for SSL Client Authentication with fallback to FORM authentication. This is *not* for using FORM based authentication over a simple SSL channel - you do not need SSL client authentication for that.

Note: Tested with Tomcat 5.5.17, 5.5.20 and 5.5.25

See also:

- SSLWithFORMFallback7
- SSLWithFORMFallback6
- SSLWithFormFallbackAuthenticator

SSL Client Authentication (sometimes also known as "Client Certificate" authentication) uses the SSL protocol to authenticate clients based on a X509 Certificate. Normally this is accomplished by configuring SSL in Tomcat, and then configuring the Web Application's security descriptor to use "CLIENT-CERT" as the auth-method in the login-config section.

We found that we wanted to implement 2 levels of security - client authentication based on SSL certificates for serious security, but FORM based login as a fallback option. This requirement can exist for a number of reasons:

- for customers who do not want to make use of certificates
- for when the customer certificate expires
- to permit customers to log in for the first time without a certificate
- to allow different "user-levels" - high security vs. low security, with different functions available
- etc...

In trying to implement this, we found the only "standard conformant" solution was to install the web application multiple times with different authentication configurations. This solution was very unsatisfactory for us, as it leads to a duplication of services, and the services are accessible under different URLs /Ports depending on the desired security level. That just wasn't what we wanted.

So the following solution, unfortunately, is not standards-conformant. This is because the J2EE standard, while deferring authentication to the container, specifies the authentication method in the webapplication deployment descriptor (web.xml). There, only one login-config section is allowed, which counts for the whole application. It does not permit you to configure a fallback login method.

Setup

So, to get the fallback login working you will need the following:

- Tomcat Installation
- Your Webapplication
- The Java Class SSLWithFormFallbackAuthenticator (download from here)
- Server Certificate & Private Key
- Client Certificate & Private Key
- Certification Authority Public Certificates
- Working authentication realm

It is assumed that your web-application is working, and you are currently using FORM based authentication. Your login config in your web.xml deployment descriptor should therefore look something like this:

```
<login-config>
          <auth-method>FORM</auth-method>
          <form-login-config>
                  <form-login-page>/pub/login.jsp</form-login-page>
                  <form-error-page>/pub/loginerror.jsp</form-error-page>
          </form-login-config>
</login-config>
```

It is further assumed that your web-application contains at least one protected page, requiring authentication, and that the login currently works using your FORM based login with an appropriate Login Realm.

Basic SSL Setup

First, setup SSL in Tomcat, as described in: http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html

Make sure basic SSL is working, without client authentication. You can test this using your browser. While you are at it, add the server's certification authority to your Browser's list of trusted certificates, if it is not there already. You should be able to access your application normally, but via the https (SSL) protocol. If you access a protected page, you should be prompted for a login using the FORM login you configured.

SSL Client Authentication

In the tomcat server.xml file, configure the server to use client authentication:

```
<Connector port="8443" minProcessors="5" maxProcessors="75"
    enableLookups="true" disableUploadTimeout="true"
    acceptCount="100" debug="0" scheme="https" secure="true";
    clientAuth="want" sslProtocol="TLS"
    keystoreFile="/etc/mykeystore.jks" keystorePass="changeit"
    truststoreFile="/etc/mytruststore.jks" truststorePass="changeit"/>
```

Note the use of clientAuth="want" to request a certificate, but not fail if none is presented.

Configure your Realm to accept the client certificate. Depending on which realm you are using, you will need to add a user for the certificate in different ways. The default Tomcat Realms use the "SubjectDN" field of the certificate as a "username" to look up the user.

Testing Client Authentication

Fire up your browser and install the client certificate and private key in your certificate store.

Now change your auth-method from "FORM" to "CLIENT-CERT" and restart/redeploy your web-app. If you access your protected page you should now be prompted for a certificate by your browser. Select the installed certificate. If everything was configured correctly you should be authenticated based on your certificate, and taken to the protected page.

If things go wrong, here's some places to look:

- Is the client certificate properly installed? If not, your browser will not offer the certificate for you to choose on login.
- Is the client certificate authority properly imported in the truststore on the server? If not, the browser will not know to use your installed client certificate.
- Is the server certificate valid and properly installed?
- Is your client certificate's SubjectDN configured as a user in the Realm you are using? Depending on which realm you are using you will have to add the user in different ways (for example, to the file "tomcat-users.xml" if using Tomcat's MemoryUserDatabase).

Adding Fallback to Form Authentication

Make sure the class you downloaded, "SSLWithFormFallbackAuthenticator.java" is compiled and installed, for example in the server/classes folder of your Tomcat installation. Alternatively, you can pack the compiled class in a JAR file, and place the JAR in the server/lib folder of your Tomcat installation.

The class implements a Tomcat "Valve", and needs to live within the server directory of your Tomcat and cannot be part of your web application's WAR, since it is used by Tomcat for authentication, which happens before your web application is called, and hence outside your web application's classloader.

Configure your Web-Application to use this class for authentication. This is done in two steps:

1. Remove the <auth-method> element from your web application's deployment descriptor. The <login-config> element should still contain the <form-login-config> elements to configure the form, but *NO* <auth-method> tag.

```
<login-config>
          <!-- auth-method is commented out for fallback authentication -->
          <!-- <auth-method>FORM</auth-method> -->
          <form-login-config>
                  <form-login-page>/pub/login.jsp</form-login-page>
                  <form-error-page>/pub/loginerror.jsp</form-error-page>
          </form-login-config>
</login-config>
```

2. Configure the authentication valve. The authentication valve can be configured in two places. Either in your Tomcat server.xml file, or your application's context.xml file. The context.xml file lives in the directory "META-INF" within your WAR file. The following is a sample context.xml file for fallback authentication:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/mycontextpath" >
        <Valve className="at.telekom.tomcat.security.SSLWithFormFallbackAuthenticator" />
</Context>
```

If you have configured the <Context> element in your Tomcat server.xml file, you can also place the <Valve> element there.

You will need to restart tomcat to apply these changes.

Testing Fallback Authentication

This is best tested with two different browsers (eg Firefox and IE):

1. Install the client certificate in one of the Browsers (if it isn't already)
2. Fire up this browser and visit your protected page
3. You should be promted for the certificate as before, select it as before
4. You should be logged into the site, as before
5. Now fire up the other browser, the one without the certificate
6. Attempt to access your protected page
7. Depending on your Browser you may be promted about the certificate - click "cancel" if this is the case
8. You should be taken to the login form of your application
9. Log in using the form. You should be granted access, as before

How does it work?

The code is tested with Tomcat 5.5.17, 5.5.20 and 5.5.25. It will probably work with only minor modifications for other Tomcat 5.5 versions. It has been tested using Java 1.5.

In short, this code works because:

- Tomcat uses the auth-config element of the deployment descriptor to create an Authentication Valve
- If this element is missing, Tomcat does not complain, and simply installs no Authenticator
- By manually adding our own Authentication Valve we add authentication back to the application
- Our authentication valve inherits from the FormAuthenticationValve, and adds functionality from the SSL authenticator
- It simply first tries SSL Auth, and failing that makes a second attempt using form auth

Comments, Feedback, Support

This code is supplied back to the apache foundation, without any support or warranty. Use at your own risk. The author and his employer assume no responsibility for damages resulting in the use of this code or these instructions.

Feel free to use the code in any way you want but do not expect support.

Should you have questions about the code, please feel free to contact me (the Author) at: runger ~~-AT-~~ aon.at

---

CategoryFAQ