NativeSPJavaInstall

Install Shibboleth to protect Java Servlets

The Shibboleth SP is presently only implemented in C++ as a module for Apache httpd, IIS, and NSAPI. However, it's quite easy to use the Shibboleth SP to provide authentication information for Java servlets in a wide variety of servlet containers.

In the setup described here, requests from browsers are intercepted first by Apache httpd. The Shibboleth SP then checks these requests to enforce authentication requirements. After an assertion is received and a Shibboleth session is established, the SP or Apache httpd can enforce access control rules, or it can just pass attributes to the application. The request is then forwarded to the servlet through the use of the AJP13 protocol. Subsequent requests can leverage the Shibboleth session or a session maintained by the application or servlet container to persist the login.

Integrating the Shibboleth SP with Grails

If you are using the Grails framework to develop Spring based Groovy/Java web applications, you can integrate your container provided Shibboleth authentication with Spring Security Core by installing the Spring Security Shibboleth Native SP plugin. The documentation is available here.

1. Setup Apache httpd with Shibboleth

Install Apache httpd first. It's by far the easiest if version 2.2 or 2.4 is used, because these versions include `mod_proxy_ajp` in the main distribution. If you're using an older version, you'll need to install `mod_jk` and set that up independently.

Next, install Shibboleth itself. This is a platform-dependent decision, so go back to the main installation page and select the right one. After you complete the installation process, please return here and continue with step 2.

2. Setup AJP13 support in your servlet container

This step depends on your servlet container.

- **Tomcat**: Tomcat has an AJP 1.3 connector enabled by default.
  - Setting the `tomcatAuthentication="false"` attribute on the AJP `<Connector>` element allows for passing `REMOTE_USER` from Apache httpd. See Tomcat's AJP Connector documentation for more.
- **Jetty**: Jetty's documentation has good instructions on how to enable both Jetty and your application to listen on AJP 1.3.

> **Jetty 9 drops AJP**
>
> Note that AJP support has been dropped starting from Jetty version 9. They recommend using `mod_proxy_http` instead of `mod_proxy_ajp`.

Be careful that there is no direct HTTP listener opened by the servlet container. If, for example, there's an HTTP connector listening on port 8080 and no interceding firewall, users would be able to directly access the servlet on port 8080, which bypasses Apache httpd. This also means they would bypass Shibboleth authentication and authorization.

AJP packet size

Service Providers that request many attributes or receive many attribute values can expect to exceed the default maximum AJP packet size (8kb). In order to prevent this, raise the maximum AJP packet size to 65kb (maximum allowed by the AJP protocol). This value should be specified **both** in Apache httpd and your servlet container configuration.

- **Tomcat**: Add a `packetSize="65536"` to the AJP `<Connector>` element.
- **Apache httpd with mod_jk**: Add a worker.<name>.max_packet_size directive to the worker definition.

```
worker.<name>.max_packet_size=65536
```

- **Apache httpd with mod_proxy_ajp**: Add a ProxyIOBufferSize directive to Apache httpd's configuration.

```
ProxyIOBufferSize 65536
```

3. Configure Apache httpd to route requests to your servlet

Add a line to your Apache httpd configuration, such as in `httpd.conf`, to map requests on the proper virtual hosts to your application through AJP 1.3.

```
ProxyPass /my-application ajp://localhost:8009/my-application
```

4. Add Shibboleth protection for your servlet

Add a line to your Apache httpd configuration on the proper virtual host, such as in `httpd.conf`, to trigger Shibboleth session initiation and authentication for your application:

```
<Location /my-application>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user
</Location>
```

Since environment variables are not passed by `mod_proxy_ajp` unless they have `AJP_` prefixes, you'll also need to add `attributePrefix="AJP_"` to the `<ApplicationDefaults>` (or appropriate `<ApplicationOverride>`) element in your shibboleth2.xml:

```
<ApplicationDefaults id="default" policyId="default"
    entityID="https://sp.example.org/shibboleth"
    REMOTE_USER="eppn persistent-id targeted-id"
    signing="false" encryption="false"
    attributePrefix="AJP_">
```

Alternatively, data can be passed via HTTP request headers (by means of using `ShibUseHeaders On`) but this is considered less secure and changes all variable names (`HTTP_NAME`, where the `NAME` is the `name` in `attribute-map.xml`).

You can then decide to enforce access control rules using shibboleth2.xml or htaccess or just use the attributes supplied in your application.

Struts 2 Issue

When deploying an application written using the Struts 2 framework, see the Java example section on the native attribute access page for an issue with retrieving attribute values with certain problematic names.