RemoteUserAuthnConfiguration

Current File(s): conf/authn/remoteuser-authn-config.xml, webapp/WEB-INF/web.xml

Format: Native Spring

Legacy V2 File(s): conf/handler.xml, web.xml

- Overview
- General Configuration
- Reference
  - Beans
- V2 Compatibility
- Notes

Overview

The authn/RemoteUser login flow relies on whatever container-based mechanism you have available (HTTP BASIC auth, LDAP, Kerberos, other SSO systems, etc.). By default, this flow is configured without support for advanced authentication controls like passive or forced authentication. In most cases, implementing support for those features in conjunction with other SSO systems should be done with the External flow.

This flow is actually implemented as a special case of the External flow that happens to use a supplied servlet to implement the External contract that supports extraction of the identity from the request. Rather than customizing this flow, it's better to leave it as is and use the External flow itself to do other things.

General Configuration

Use authn/remoteuser-authn-config.xml and/or edit-webapp/WEB-INF/web.xml to configure this flow.

The shibboleth.authn.RemoteUser.externalAuthnPath bean defines the flow redirection path to the resource that's used to pick up the container-established identity, by default a context-relative location. It can be modified if needed.

As of V3.3, if you need to support multiple external paths, you can supply a bean of type Function<ProfileRequestContext,String> with the name shibboleth.authn.RemoteUser.externalAuthnPathStrategy.

The shibboleth.authn.RemoteUser.ClassifiedMessageMap bean is a map of error messages to classified error conditions that isn't often used with this handler because it generally won't return with any contextual details that you can make use of. If this comes into play, the External flow is probably what you're meant to use.

The servlet that implements the authentication for this flow is configured in webapp/WEB-INF/web.xml and can be given parameters that extend it to support arbitrary headers or request attributes instead of or in addition to the REMOTE_USER reserved CGI variable (the identity of the requester as established by the web server / container, which Java exposes in the Servlet API with the getRemoteUser method). By default, it supports only the "real" REMOTE_USER value exposed via that API (this does not include headers named to emulate it, a very poor practice due to the potential for confusion).

As always, when editing this file, make sure to copy it to edit-webapp/WEB-INF/web.xml first and make any changes to that copy.

The following servlet init parameters are supported:

- checkRemoteUser (true or false)
  - whether to look for a principal name in REMOTE_USER
- checkAttributes (list of space-delimited request attribute names)
- checkHeaders (list of space-delimited request header names)
- subjectAttribute (name of request attribute)
  - single request attribute to check for a Java Subject to use for the authentication result
- authnMethodHeader (name of request header)
  - request header(s) to check for "method" strings to attach to Java subject as custom principals
- authnAuthorityHeader (name of request header) (V3.4+)
  - request header(s) to check for URIs to attach to Java subject as proxied authenticating authorities

The algorithm of the servlet is roughly:

1. If subjectAttribute is set, check it for a Java Subject to use. If found, it will be returned through the External Authentication interface and all the other settings are ignored.
2. Otherwise, check for a principal name as directed by the settings, in REMOTE_USER, attributes, and headers (in that order). If not found, authentication fails.
3. If authnMethodHeader is set, check each value to see if the associated login flow supports a custom Principal matching the value in the header, and if so, attach that Principal to the Subject returned through the External Authentication interface. Note that if the header contains a value not supported by the associated login flow, it will be logged, but otherwise not impact the success of this flow. By the time this feature is executing, it's already a given that the result was successful.
If you need something different from this, your best option is to use the External login flow and possibly adapt the existing servlet as an example to copy from for your own purposes. In all respects, this flow is simply a more concrete use of the External flow.

Reference

Beans

The beans defined in `authn/remoteuser-authn-config.xml` follow:

<table>
<thead>
<tr>
<th>Bean ID</th>
<th>Type</th>
<th>Default</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>externalAuthnPath</td>
<td></td>
<td>/RemoteUser</td>
<td></td>
</tr>
<tr>
<td>shibboleth.authn.RemoteUser.</td>
<td>Function&lt;ProfileRequestContext, String&gt;</td>
<td></td>
<td>A function that returns the redirection expression to use for the protected resource.</td>
</tr>
<tr>
<td>externalAuthnPathStrategy</td>
<td>3.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shibboleth.authn.RemoteUser.</td>
<td>Map&lt;String, List&lt;String&gt;&gt;</td>
<td>various</td>
<td>A map between defined error/warning conditions and events and</td>
</tr>
<tr>
<td>ClassifiedMessageMap</td>
<td></td>
<td></td>
<td>implementation-specific message fragments to map to them.</td>
</tr>
<tr>
<td>shibboleth.authn.RemoteUser.</td>
<td>Predicate&lt;ProfileRequestContext&gt;</td>
<td></td>
<td>An optional bean that can be defined to control whether to preserve</td>
</tr>
<tr>
<td>resultCachingPredicate</td>
<td></td>
<td></td>
<td>the authentication result in an IdP session</td>
</tr>
<tr>
<td>shibboleth.authn.RemoteUser.</td>
<td>Boolean</td>
<td>true</td>
<td>Whether to add the content of the supportedPrincipals property of the</td>
</tr>
<tr>
<td>addDefaultPrincipals</td>
<td>3.2</td>
<td></td>
<td>underlying flow descriptor to the resulting Subject</td>
</tr>
<tr>
<td>shibboleth.authn.RemoteUser.</td>
<td>Pattern</td>
<td></td>
<td>Regular expression to match username against</td>
</tr>
<tr>
<td>matchExpression</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

V2 Compatibility

This flow is designed to be integration-compatible with the V2 RemoteUser login handler in that it operates by default at the same context-relative path of /Authn/RemoteUser. Any web server or `web.xml`/authentication configuration should work unmodified (but of course any `web.xml` configuration needs to be applied to the V3 file copied to `edit-webapp/WEB-INF`).

Notes

This flow is configured by default without support for non-browser profiles (namely ECP) because the RemoteUserInternal flow is a better choice for container-based authentication when a browser isn't required. It eliminates the extra redirects used by this flow. If your clients can handle the redirects and you prefer to use this flow, you can change or remove the `nonBrowserSupported` property set on the flow in `authn/general-authn.xml`.