Using a Tomcat Realm for external authentication

Background

VIVO is not written to use the standard JEE or Tomcat authentication systems, so using a Tomcat Realm would require some customization. This doesn't seem very difficult, it just hasn't been a priority for us.

When VIVO is set up to use external authentication, it uses a reverse-proxy setup, where an Apache HTTP server intercepts all calls to Tomcat. The Apache server uses a Shibboleth module or other module to secure a particular page: http://localhost:8080/vivo/loginExternalAuthReturn.

If an HTTP request is made to that page, and the request does not belong to a session that is already logged it, the Shibboleth module in Apache will intercept the request and guide the user through the authentication process. When the user's credentials are accepted, the module invokes the secured page, as requested, storing the user's ID in one of the HTTP headers. The VIVO code reads the user ID from the HTTP header and stores it in the session object. Only that one page is secured, and VIVO remembers the user ID for use in subsequent requests.

Which HTTP header will VIVO inspect for the user ID? The header which is named in externalAuth.netIdHeaderName.

Most institutions that use VIVO also use Shibboleth in their web applications, or something with a similar mechanism. The IT group at the institution provides the VIVO implementers with the appropriate Apache module and configuration information.

I don't know of anyone who has tried to use a Tomcat Realm to accomplish external authentication in VIVO. I think it would require some small modification of the VIVO code, perhaps a change to ExternalAuthHelper.getExternalAuthId(). Tomcat would use the Realm to create a Principal object in the HTTP request, and VIVO would get the user ID from that Principal instead of looking in an HTTP header. Web.xml would be modified to secure the page, as you have already done.

Testing

It really was just that easy!

I added these lines to ExternalAuthHelper.getExternalAuthId(), right after the check for a null request object:

```
Principal p = request.getUserPrincipal();
if (p != null) {
   log.debug("Found a UserPrincipal in the request: " + p);
   String userId = p.getName();
   if (StringUtils.isNotEmpty(userId)) {
       log.debug("Got external auth from UserPrincipal: " + userId);
       return userId;
   }
}
```

I added these lines to the end of web.xml, just before the closing </web-app>:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>ExternalAuthPage</web-resource-name>
        <url-pattern>/loginExternalAuthReturn</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>tomcat</role-name>
    </auth-constraint>
</security-constraint>

<login-config>
    <auth-method>BASIC</auth-method>
</login-config>
```

I set this property in deploy.properties:

```
externalAuth.buttonText = Log in using basic Tomcat
```

And voila, my tomcat-users.xml file is my external authentication system!

Obviously, you will want to use FORM authentication, instead of BASIC, and something other than the default Realm. But I expect you know how to do that already.

Please, let me know how this progresses for you. This may be something that we will add to the next release.

Jim Blake