Security - Authentication and Authorization

Fedora relies on the servlet container it is running in (Tomcat, Jetty, etc.) to authenticate the user. However, authorization is handled by Fedora using access control lists (ACLs) stored in the repository and defined using the Web Access Control (WebAC) vocabulary.

Fedora uses Apache Shiro to configure its authentication and authorization processes.

Overview

Incoming requests to Fedora first pass through a series of authentication and authorization servlet filters. These filters consult the current Shiro security manager to check whether the current user is authenticated, and then what WebAC permissions they have with regards to the requested resource. Once those permissions have been determined, the WebAC filter will either reject the request with a "403 Forbidden" response, or allow it to pass on through to be processed by Fedora.

Additional principal provider filters can also be added to this filter chain to do additional authorization processing. For instance, in the case where there is an external authentication system like Shibboleth that adds the user's security principals as an HTTP header to the request, you can configure the HTTP Header Principal Provider to extract the relevant principals and add them to the current user.

These Shiro components connect with Modeshape's authentication/authorization system through the ShiroAuthenticationProvider and ShiroSecurityContext, which implement the Modeshape AuthenticationProvider and SecurityContext interfaces, respectively. These classes do little more than delegate the questions of user identity and authentication status to Shiro and the configured realms.

Security Manager

Fedora uses an instance of the DefaultWebSecurityManager as its security manager. This manager is configured with two realms: ServletContainerAuthenticatingRealm and WebACAuthorizingRealm.

ServletContainerAuthenticatingRealm

This realm takes a ContainerAuthToken, which holds a user principal and list of role names of a user that has been authenticated by the servlet container, and adds those as principals to the current Shiro user.

Normally, a Shiro authenticating realm would consult some external data source (e.g., a database, LDAP directory, etc.) to perform the actual authentication of the user. However, the way Fedora is currently configured, those checks are all handled by the servlet container, so any authentication request to this realm will return success.

WebACAuthorizingRealm

This realm determines the permissions that the current user (who may be anonymous) has with regards to the requested resource. These permissions are represented as WebACPermission objects, which consist of a pairing of an access mode (e.g., acl:Read, acl:Write) with a resource URI.

Filters

Shiro handles customization of authentication and authorization by using servlet filters. Fedora is configured with two such filters: ServletContainerAuthFilter and WebACFilter.

ServletContainerAuthFilter

This filter checks the incoming servlet request to see if there is a user principal (indicating that there is a user that has authenticated with the servlet container). It also checks if that user has either of the roles **fedoraAdmin** or **fedoraUser**. It combines the servlet user name and the role name into a ContainerAuthToken, which it uses to "log in" to the ServletContainerAuthenticationRealm. The actual credential verification has already taken place in the servlet container; this "logging in" is just to inform Shiro that there is an authenticated user, and what their name and roles are.

WebACFilter

This filter does the main work of allowing or prohibiting requests. Based on the HTTP method (and possibly other details in the headers or body of the request) of the incoming request, and the set of WebACPermission objects that the WebACAuthorizingRealm has determined for the current user, this filter will either reject the request with a "403 Forbidden" HTTP response, or allow the request to continue on to the Fedora servlet.

Configuration

Spring configuration

```
<bean name="modeshapeRepofactory"
    class="org.fcrepo.kernel.modeshape.spring.ModeShapeRepositoryFactoryBean"
    p:repositoryConfiguration="${fcrepo.modeshape.configuration}"
    depends-on="authenticationProvider"/>

<bean name="authenticationProvider" class="org.fcrepo.auth.common.ShiroAuthenticationProvider"/>

<!-- *************************
          Authentication
     ************************* -->

<!-- Optional PrincipalProvider filter that will inspect the request header, "some-header", for user role
values -->
<!--
<bean name="headerProvider" class="org.fcrepo.auth.common.HttpHeaderPrincipalProvider">
    <property name="headerName" value="some-header"/>
    <property name="separator" value=","/>
</bean>
-->

<!-- Optional PrincipalProvider filter that will use container configured roles as principals -->
<!--
<bean name="containerRolesProvider" class="org.fcrepo.auth.common.ContainerRolesPrincipalProvider">
  <property name="roleNames">
    <util:set set-class="java.util.HashSet">
      <value>tomcat-role-1</value>
      <value>tomcat-role-2</value>
    </util:set>
  </property>
</bean>
-->

<!-- delegatedPrincipleProvider filter allows a single user to be passed in the header "On-Behalf-Of",
       this is to be used as the actor making the request when authenticating.
       NOTE: Only users with the role fedoraAdmin can delegate to another user. -->
<bean name="delegatedPrincipalProvider" class="org.fcrepo.auth.common.DelegateHeaderPrincipalProvider"/>

<bean name="accessRolesProvider" class="org.fcrepo.auth.webac.WebACRolesProvider"/>

<!-- Shiro Auth Confiuration -->
<!-- Define the Shiro Realm implementation you want to use to connect to your back-end -->
<!-- WebAC Authorization Realm -->
<bean id="webACAuthorizingRealm" class="org.fcrepo.auth.webac.WebACAuthorizingRealm" />

<!-- Servlet Container Authentication Realm -->
<bean id="servletContainerAuthenticatingRealm" class="org.fcrepo.auth.common.
ServletContainerAuthenticatingRealm" />

<!-- Security Manager  -->
<bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
  <property name="realms">
    <util:set set-class="java.util.HashSet">
      <ref bean="webACAuthorizingRealm"/>
      <ref bean="servletContainerAuthenticatingRealm"/>
    </util:set>
  </property>
  <!-- By default the servlet container sessions will be used.  Uncomment this line
      to use shiro's native sessions (see the JavaDoc for more): -->
  <!-- <property name="sessionMode" value="native"/> -->
</bean>

<!-- Post processor that automatically invokes init() and destroy() methods -->
<bean id="lifecycleBeanPostProcessor" class="org.apache.shiro.spring.LifecycleBeanPostProcessor"/>

<!-- Authentication Filter -->
<bean id="servletContainerAuthFilter" class="org.fcrepo.auth.common.ServletContainerAuthFilter"/>

<!-- Authorization Filter -->
<bean id="webACFilter" class="org.fcrepo.auth.webac.WebACFilter"/>

<bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
```

```
  <property name="securityManager" ref="securityManager"/>
  <property name="filterChainDefinitions">
    <value>
      <!-- The Auth filter should come first, followed by 0 or more of the principal provider filters, -->
      <!-- and finally the webACFilter -->
      /** = servletContainerAuthFilter,delegatedPrincipalProvider,webACFilter
    </value>
  </property>
</bean>
```