OpenID

OpenID overview

OpenID is an open specification for distributed authentication for web apps popularly used for social networking applications. Users are identified by a URL or XRI. This leads to information about where the user's "ownership" of the identifier can be validated. Such a web site is called an "OpenID Provider" (OP). A web site using OpenID to authenticate users is a "Relying Party".

A Relying Party website will have a form field for the user to enter their OpenID identifier. The relying party web app can then identify the OP the user is known to and through a series of indirect (redirects through the user's user agent) and direct http/https requests to the OP establish whether the user is in fact authenticated to the OP's standards. Typically the OP will use something like username/password authentication to authenticate the user. Furthermore typically an OP will notify the user that a Relying Party is requesting authentication.

What does a Relying Party get?

After successful OpenID authentication, the Relying Party gets a identifier string for the user. Its possible to transfer additional profile information such as email address, but this is not readily usable for authorization decisions. There's also the identity of the OP. So there are about 2.5 pieces of information available:

1. user id
2. OP id
3. the fact that the user is authenticated by some OP
   For JavaEE security these need to be mapped to application specific roles. This means that either there is very coarse grained security based only on the fact that the user is authenticated or on which OP they are using, or the relying party web app needs explicit knowledge of each user to map them to appropriate roles.

Geronimo support

Geronimo OpenID support uses the openid4java libraries.

OpenID Provider support

The `org.apache.geronimo.plugins/openid-provider-jetty//car` plugin is a simple OP app for jetty. Currently it uses a simple properties file login module for user authentication. This needs to be moved to a separate plugin so the realm can be pluggable. The application uses form based login so it maintains a session for the authenticated user: thus for repeated requests from relying parties only the "do you want to allow authentication to this relying party" notice is necessary, not actual app login for every request.

When an authentication request from a relying party arrives (through a redirect through the user's agent(browser)), the interaction is as follows:

- if the user is not authenticated (request.getUserPrincipal is null) then the user is redirected (through the user agent) to the (protected) authentication page. Through normal Java EE security this results in requiring the user to log in before getting to the authentication page.
- if the user is already logged in the request is forwarded (through a request dispatcher) to the authentication page.
  If the user accepts the authentication request from the relying party, the openid protocol continues.

The OpenID provider appears to work at least minimally based on modified unit tests from the openid4java project.

> ⚠ Currently the OP accepts authentication requests from any Relying Party. Possibly this should be restricted to either authorized RPs or RPs not on a "blacklist".

Relying Party support

The OpenID protocol specifies a sequence of message exchanges between the relying party and the user agent (and thence the OP, through redirection). It is thus a good candidate for a JASPI module, substituting for the standard basic, digest, or form based message exchanges. This is being developed in the jaspi component. When complete it will allow specifying OpenID authentication just as you can now specify basic, digest, or form authentication, although not through web.xml.