Java Library for Proxy Authenticating to WSPs

Requirements for this library

(Copied and extended from parent page).

- Need an HTTP client API allowing access to WSPs while handling authentication internally
- Client stack needs support for TLS client and server authentication, cookies, and whatever features WSPs would impose (sensible following of redirects, e.g.)
- Internals must support detection of ECP requests, Liberty-defined interactions with IdP, and handling ECP responses

Starting points

Links and notes about available Java software on which to build Java library for making use of proxy authentication via SAML.

java-openws

Internet2 Middleware / Shibboleth have a Java library for making use of Web Services that is intended for use in the Shibboleth IdP (which is implemented in Java). This is the most obvious and immediately applicable starting point.

(Note that there's a REL_1 branch that's apparently the right place to be, not trunk.)

Apache / Jakarta / Commons / HttpClient 3 / HttpComponents HttpClient 4 beta

HttpClient 3

(See also website).

The obvious choice here is HttpClient 3 and indeed java-openws depends upon HttpClient 3.

The tremendous upside of using this library is that it is quite stable, mature, and well-adopted. The downside of using this library is it doesn't appear to be under continued development on the 3.x line. Reassuringly, the Http Components project states a commitment to maintain HttpClient 3 until 4 is deemed ready to supersede it.

HttpClient 3 has extension points for Authentication.

HttpClient 3 handles cookies and SSL (both client and server TLS, but may need customized to cope with custom key storage?).

HttpComponents HttpClient 4

(See also website).

HttpClient has been re-architected and re-written under the Apache HttpComponents project and there's a beta2 release available of HttpClient 4. This would be a more comfortable library to pull in if it were GA released but it is still worth looking at.

Relevant features:

- Supports encryption with HTTPS (HTTP over SSL) protocol.
- Basic, Digest authentication schemes. Please note NTLM is supported only partially.
- Plug-in mechanism for custom authentication schemes.
- Pluggable secure socket factories, making it easier to use third party solutions
- Plug-in mechanism for custom cookie policies.
- Direct access to the response code and headers sent by the server.

JASIG Web Proxy Portlet

(See wiki page).

This, and subsequent Eric Dalquist thoughts and directions as this portlet marches towards inclusion in uPortal 3.1, is a decent starting point for thinking about the necessary caching and redirect-following behaviors.

WSS4J

Of course, the proposed Java library needs to be doing more than just HTTP – HTTP is the transport layer for the security interactions that are being coded here. One somewhat disparaged starting point for WSS implementation is WSS4J.

http://ws.apache.org/wss4j/

Design Issues

Finding the IdP

During processing, the library (acting as an ECP client) needs to be able to route `<AuthnRequest>` messages to the logged-in user's IdP. The identification of the IdP's entityID (it's unique name) is a simple matter of obtaining the value in the "Shib-Identity-Provider" request header the SP supplies.

Once obtained, the ECP client is then obligated to ensure that if the WSP's request includes a SOAP header identifying the set of allowable IdPs then the user's IdP is in that list. Failing that, the ECP client is charged with immediately returning a SAML error response that it creates itself and returning it to the SP at the location designated in another SOAP header.

Otherwise, the ECP client has to take the step of identifying **where** at the IdP it should send the WSP's request message, and how it should authenticate itself to that IdP. In ID-WSF terms, this means identifying the Single Sign-On Service endpoint at the IdP and the "security mechanism" it needs to use.

We have settled on the following approach of carrying an EndpointReference in a SAML attribute supplied in the SSO assertion.

The challenge here is that the IdP has to be able to generate that XML-valued attribute (not too complex, probably), but then the information has to be available to the client library. This means the library either needs to parse into the assertion to some degree and access the information directly, or the SP needs to do so on its behalf, leaving open how that information could be "flattened" into one or more headers in a sensible way.