Shibboleth SSO for small business

August 14, 2012 / Gavin Jackson

java linux shibboleth SSO web

Recently my team completed a Single Sign On (SSO) project - we used an Open Source SSO implementation called Shibboleth. In total it took about 8 weeks to get this into production.

Rather than list config files and install processes (which you can find online), I thought it might be a more interesting post to discuss the methodology that was used and some of the interesting hurdles that were encountered.

Caveat is that this is a pretty basic configuration, only one IdP within a single enterprise, Shibboleth has been designed to scale out far beyond our little implementation (to support SSO across academic institutions), this is referred to as federation but is beyond the scope of this article.

Shibboleth consists of two main components - an identity provider (IdP) and one or more service providers (SPs). These components use the Security Assertion Markup Language (SAML v2) to obtain and reuse authentication and user attribute information (for example name, address, group roles etc).

The IdP is responsible for authentication and attribute resolution and the SP is responsible for protecting and granting access to web resources.

The IdP and SP need to have knowledge of each other - this is required for the SP to redirect the user to the IdP for authentication, establish crypto to facilitate message signing and encryption between the two servers and to define what attributes can be released from the IdP to specific SPs.

This is done via xml metadata files, it is really important to understand how these are generated and where they are referenced to have a working configuration. Step 1: Get something working.

The first POC was to get the IdP talking to an authentication service, fortunately our organisation has an Active Directory server that uses the built-in ldap authentication module. First step was to set up an IdP that talks to AD.

Step 2: Integrate with our customer authentication database. Shibboleth can easily plug in external authentication providers using JAAS, I wrote a small plugin that authenticates users against our application server. Step 3: Obtain customer attributes.

Shibboleth supports two main methods for retrieving attributes - via a JDBC connection to a database or via LDAP. The first cut used an sql query to get these permissions, a later version used a custom attribute resolution plugin to get these attributes via calls to our application server (this was an architectural decision).

Step 4: Configure and test with multiple SPs, this is where we start seeing shibboleth start to shine - there is something really cool about reusing the authenticated session across physically seperate web servers, this POC provided us withe some assurance that this SSO implementation could work on an off site web server.

Step 5: Get logout working. Sounds simple right This probably deserves a post of its own, but a week was spent figuring out how to log out of the SSO session.

Step 6: Update our website to provide login button, logged in menu and logout button. mod_proxy and mod_ajp were used to proxy requests from apache to our existing tomcat web apps (and define shibboleth security restrictions based on log in status and specific user attributes).

The user attributes are made availabkle to php via request attributes, and we have verified that these can be accessed by python via mod_wsgi. Step 7: Customise the IdP login and error pages.

Today I configured an offsite Linode instance (hosted in Tokyo), Shibboleth SP that talks to our IdP here in Canberra - works flawlessly (after performing some careful configuration and generating/wiring up the required SP and IdP metadata.

Hope this article helps highlight some of the complexity involved with rolling out an SSO implementation and demostrates how a phased implementation can help get you over the finishing line - you will need to spend time reading shibboleth doco, newsgroups and partake in mailing lists - I don't think there is an easy way around this at present.