NCState EZProxy Case Study

Background

NCSU Libraries have been using EZProxy as its only method of providing off-campus access to electronic resources for many years. Currently, the authentication process involves a number of CGI scripts, all of which have been modified over the years, and as programmers and systems staff have come and gone the understanding of how all of these scripts work together has mostly been lost. In some places there are still pieces of code that had been included to work around one resource or another. At some point in time it was decided that, due to licensing issues, we needed some method to restrict our Friends of the Library, who receive limited access to electronic resources, from accessing specific resources. And so a hack was included to check the requested resource and compare it to a "blacklist" of restricted resources.

Earlier this year the Office of Information Technology (central IT) put together a working group to investigate and implement a campus-level Shibboleth Federation and the library was included in this group.

Because EZProxy natively supports Shibboleth, this was the perfect time to begin work on new proxy architecture. The biggest hurdle in this was to come up with a method to include our Friends of the Library (FoL) patron database. In the current system, FoL members are given an account in the Library ILS system (SirsiDynix Symphony) and a second authentication path had been created to work in parallel with the normal campus authentication. In this way, it works very similar to a Shibboleth DS/WAYF in that users select their NCSU Library affiliation and are taken to a different form to enter their library account login credentials. This information is used to look them up in the Symphony database and grant them access.

Due to the way the FoL accounts are created and the workflow that was already in place, it was not going to be possible to dump out the user accounts and import them into a standard directory server like LDAP or Active Directory.

Implementation

Shibboleth can use an RDBMS to resolve attributes and contains a built-in connector for querying an external database. However, Shibboleth does not have a native JAAS login module for using a database as your directory source. A JAAS RDBMS login module was created to use the Symphony database as the basis for authentication. A significant challenge for a non-programmer; however, there are good examples for JAAS on the Internet and there were only a couple of major changes from the examples, the database connection strings and the actual query used for authentication. Once this was built, the new login jar file, along with the necessary database drivers were added to the IdP build and thoroughly tested.

With a working IdP authenticating FoL members to EZProxy, the next step was to create a few attributes that EZProxy could use as a basis for groups or potentially statistics gathering. The Symphony database leaves much to be desired for user accounts; the name field for example, is a single field rather than separate fields for first and last name. Compounding the issue is the way data has been entered, i.e. First Name, Last Name. In the end, a number of attributes were extracted in addition to patron name; an expiration date, the users account number, email, if available, and an "affiliation". The patron account system in Symphony actually contains an account for everyone on campus plus system accounts, ILL accounts, etc. A subset of these accounts have a different prefix and include other types of users in addition to FoL members. The affiliation attribute may be useful later on if we need to group resources ever further.

Test accounts were created with various affiliations and more testing was done to ensure the IdP was working correctly with EZProxy. The next steps were adding the library IdP to the campus federation and creating the campus Discovery Service, which were done by the campus Shibboleth administrator.

The final step was configuring EZProxy to use the Discovery Service. From the EZProxy documentation, this should be straightforward, however, there is a bug in the EZProxy application that generates the default metadata for EZProxy. If it isn't a bug, then it is a serious omission in the EZProxy documentation.

The only specific directive given in the documentation regarding EZProxy's metadata is to add the EntityID to the EntityDescriptor. However, a few entries in the default metadata were erroneous preventing the Discovery Service from properly connecting to EZProxy, necessitating additional editing that is not documented.

Within the EntityDescriptor is the SPSSODescriptor that contains the string denoting the various levels of SAML support. EZProxy was generating this string containing a superfluous reference to a namespace. Additionally, in the DiscoveryResponse entry, while the binding string was created correctly, the location parameter again had the reference to a namespace instead of the EZProxy server's Discovery Service endpoint. Once these items were set correctly and the metadata updated on the campus side, we were then able to use EZProxy against the Discovery Service.

Default Metadata

```
<md:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol http://schemas.xmlsoap.org/ws/2003/07/secext">
<md:Extensions>
<idpdisc:DiscoveryResponse Binding="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol" Location="urn:oasis:names:tc:SAML:2.0:protocol http://schemas.xmlsoap.org/ws/2003/07/secext" index="1" xmlns:idpdisc="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol"/>
</md:Extensions>
```

Updated Metadata

```
<md:SPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol urn:oasis:names:tc:SAML:1.1:protocol">
<md:Extensions>
<idpdisc:DiscoveryResponse Binding="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol" Location="https://proxy.lib.ncsu.edu/Shibboleth.sso/DS" index="1" xmlns:idpdisc="urn:oasis:names:tc:SAML:profiles:SSO:idp-discovery-protocol"/>
</md:Extensions>
```

Planned Use Flow

The architecture is still undergoing testing, however, the use flow should be set at this point and a diagram describing this flow is attached. The resources were configured in EZProxy and groups were created. To minimize the level of complexity with the EZProxy grouping system, two main groups were created, a restricted group and an unrestricted group. Keeping with the simplicity scheme, EZProxy was then configured to allow users authenticating against the Campus IdP access to both restricted and unrestricted resource groups and users going through the Library IdP access to only the unrestricted resource group. Ultimately, some additional decision points will be made, e.g. an expired account notification, other groups based on use of affiliation attribute, etc.

A concern remains regarding usage statistics for electronic resources. Under the CGI authentication scheme we are able to capture access to electronic resources in our web server logs. With the planned use flow, user requests will occur only on the proxy server meaning that we will lose information regarding local use. In the flow diagram, EZProxy is configured normally to ignore requests from local IP addresses. It's likely that in order to track electronic resource usage, we may force all access through proxy.

Error rendering macro 'gliffy'

java.lang.IllegalArgumentException: Content must not be null