Testing ezproxy workflow with EBSCO Interface

Testing of proposed workflow via ezproxy and Shibboleth SessionInitiators

David Kennedy

University of Maryland

August 1, 2007

Abstract

In this experiment, I tested the workflow, which was put forth by the InCommon Library Services Working Group, for seamless authentication into Shibboleth-protected licensed library resources. The workflow is designed to handle links generated from many places, including OpenURL resolvers, gateways to online resources, class pages in Blackboard, faculty-created links, etc. To be considered successful, these links need to be relatively easy to generate.In brief, the workflow involves links generated to the licensed resource through a URL rewriting proxy, the URL rewriting proxy redirecting those links to SessionInitiator at the licensed resource, and the licensed resource seamlessly redirecting to the proper institution's Identity Provider for authentication.For this experiment, I tested with ezproxy BETA 4.2a, shibboleth identity provider v. 1.2.1a for Universityof Maryland, College Park, and EBSCO as licensed resource/Service Provider. I also tested links generated from an OpenURL resolver, SFX, and from a gateway to library resources, Metalib.

Test Scenarios

For the purpose of this experiment, I isolated each piece of the workflow separately, and eventually tested ezproxy and EBSCO in a real library environment. Below are the steps I took in testing:

- Test Session Initiators
- Test ezproxy SPUEdit directives
- Test ezproxy SPUEdit directives and Session Initiators in a controlled environment
- +Test EBSCO's Session Initiators+
- Test ezproxy SPUEdit directives configured for EBSCO Session Initiator
- Test ezproxy/EBSCO in a real library environment

Methods

Test Session Initiators

I installed a 1.3f Service Provider on biblos.umd.edu. The service provider was configured with two endpoints, /cgi-bin/printenv and /secure, and one SessionInitiator, /Login. I configured the Service Provider to trust three identity providers, urn:mace:incommon.umd.edu, urn:usmai:cp:logintest.lib.umd.edu, and urn:usmai:hs:logintest.lib.umd.edu.I wanted to create URLs that would eventually resolve to the endpoints, but would force authentication to the proper IdP, without explicitly asking the user to identify where they were from.
Some of the links I used:

https://biblos.umd.edu:4243/Shibboleth.sso/Login?providerId=urn:usmai:cp:logintest.lib.umd.edu&target=https://biblos.umd.edu:4243/cgi-bin/printenv

https://biblos.umd.edu:4243/Shibboleth.sso/Login?providerId=urn:mace:incommon:umd.edu&target=https://biblos.umd.edu:4243/secure

SessionInitiators behaved as expected, redirecting through proper IdP and on to proper endpoint after authentication.

Test ezproxy SPUEdit directives

I tested the SPUEdit directives for a) that they redirected properly according to the regular expressions, b) that the session was not proxied, and c) that ezproxy did not force authentication. I configured ezproxy's shib.usr to deny all proxying. I configured a few simple SPUEdit directives. I tested these and verified that I was redirected to the proper URLs, my session was not proxied with ezproxy, and I was not prompted to log in.

Further, I tested SPUEdit IP directives in order for ezproxy to selectively redirect on-campus and off-campus traffic. Specifically, I tested creating an on-campus range of IPs with ExcludeIP directives, and creating SPUEdit -ExcludeIP directive. Tested the behavior within and outside the defined range.

Example:

ExcludeIP 129.2.16.108-129.2.16.108
SPUEditVar provider=urn:usmai:cp:logintest.lib.umd.edu
SPUEdit -ExcludeIP @^https?://.**\.ebscohost\.com/.**$@http://abc.com/target=$0@irs
SPUEdit @^https*://.**\.ebscohost\.com/.**$@http://def.com/target=$0@irs
Option LogSPUEdit

Test ezproxy SPUEdit directives and Session Initiators in a controlled environment

In this step, I used the Service Provider and ezproxy from the previous two steps. The service provider was running at https://biblos.umd.edu:4243, and the ezproxy was running at http://biblos.umd.edu:2048. The goal here was to create simple ezproxy urls to get to my shibboleth-protected endpoints, and force authentication to the correct IdP seamlessly.

I chose two endpoints:
https://biblos.umd.edu:4243/secure
https://biblos.umd.edu:4243/cgi-bin/printenv

I configured SPUEdit in ezproxy.cfg to use the SP's SessionInitiator. In our environment, as I suspect in most environments, a single ezproxy instance serves one institution, and so I hard-coded the Identity Provider's providerId into the SPUEdit directive. I tested with three different providerIds.

SPUEdit @^https://biblos\.umd\.edu:4243/.*$@https://biblos.umd.edu:4243/Shibboleth.sso/Login?providerId=urn:usmai:cp:logintest.lib.umd.edu&target=$e0@irs

Then, I created simple ezproxy urls to reach my endpoints:

http://biblos.umd.edu:2048/login?url=https://biblos.umd.edu:4243/secure
http://biblos.umd.edu:2048/login?url=https://biblos.umd.edu:4243/cgi-bin/printenv

Test EBSCO's SessionInitiators

In this stage, I tested Shibboleth authentication to EBSCO resources via Session Initiator that EBSCO configured into their Service Provider. I tested linking to an online database, Academic Search Premier, and linking directly to a Full Text Journal, Journal of Financial and Quantitative Analysis. I configured the IdP to release eduPersonScopedAffiliation. I edited our EBSCO account to map valid Affiliations to our account.

I tested with the following URLs:
(Academic Search Premier)
https://shibboleth.ebscohost.com/Shibboleth.sso/Login?providerId=urn:usmai:cp:logintest.lib.umd.edu&target=http%3A//shibboleth.ebscohost.com/login.aspx%3Fauthtype%3Dshib%26profile%3Dehost%26defaultdb%3Daph
(Journal of Financial and Quantitative Analysis)
https://shibboleth.ebscohost.com/Shibboleth.sso/Login?providerId=urn:usmai:cp:logintest.lib.umd.edu&target=http%3A//search.ebscohost.com/direct.asp%3Fdb%3Dbuh%26jn%3DFQA%26scope%3Dsite

EZproxy and the SP were able to resolve to the proper endpoints and provide the proper authentication without user interaction.

Test ezproxy SPUEdit directives configured for EBSCO Session Initiator

I configured ezproxy to take advantage of EBSCO's SessionInitiators in order to create more simple links to the above endpoints, and maintain the same functionality (seamless authentication). I configured the following SPUEdit directive in ezproxy:

SPUEdit @^https*://.**\.ebscohost\.com/.**$@https://shibboleth.ebscohost.com/Shibboleth.sso/Login?providerId=urn:usmai:cp:logintest.lib.umd.edu&target=$e0@irs

I then tested the following ezproxy urls:
http://biblos.umd.edu:2048/login?url=http://shibboleth.ebscohost.com/login.aspx?authtype=shib&profile=ehost&defaultdb=aph
http://biblos.umd.edu:2048/login?url=http://search.ebscohost.com/direct.asp?db=buh&jn=FQA&scope=site

Test ezproxy/EBSCO in a real library environment

In this stage, I tested all of the pieces together by installing the ezproxy instance into an environment where it was integrated with a library catalog (ALEPH), an OpenURL resolver (SFX), and a gateway/metasearch tool (Metalib). I configured this ezproxy instance with the same SPUEdit directive as in previous test. I also modified the database links in metalib to reflect the new login syntax.

For instance, I changed the Business Source Premier URL from

http://search.ebscohost.com/login.aspx?authtype=ip,uid&defaultdb=buh&profile=ehostto

http://shibboleth.ebscohost.com/login.aspx?authtype=ip,shib&defaultdb=buh&profile=ehost

Both SFX and Metalib are aware of ezproxy and know how to generate URLs through ezproxy. They are also both aware of a user's IP address and can be configured to recognize when a user is on-campus as opposed to being off-campus. In our environment, Metalib and SFX redirect on-campus users directly to online resources and off-campus users through ezproxy. So, besides the URL changes to the EBSCO databases, no other changes were necessary to SFX or Metalib.

Tested accessing EBSCO resources through Metalib and SFX. All tests were successful.

Conclusions

The goal of this experiment was twofold: 1) to test a method for providing seamless authentication to library-licensed online resources, and 2) to test that this method would integrate with existing library technology and allow relatively easy creation of authenticated URLs to library-licensed online resource. In both cases, this experiment was successful.

Using EBSCO online resources as an example, we were able to take a user from an OpenURL resolver or online gateway directly to the resource without ever asking the user to identify where they were from or ask the user to log in more than once. And, so we were able to provide seamless authentication to resources.

For the second part of the goal, we tested SFX and Metalib as existing library technology. SFX and Metalib are not unique in their category of software in that they are aware of ezproxy, and that they can act according to a user's location. Very little, if any, configuration needed to be done to the existing library software in order to provide access to EBSCO online resources.

In terms of ease of URL creation, a user, say a faculty member, could take a known URL for an online resource, and simply tack an ezproxy prefix (ex. http://biblos.umd.edu:2048/login?url=) to the beginning of the known URL.

Recommendations

For Resource Providers:

- Implement SessionInitiators to avoid the WAYF when at all possible
- If a resource provider is going to provide multiple forms of authentication, specifically IP and Shibboleth based authentication, there should be a single URL syntax that works for both methods.

For Library Software Admins:

- Use ezproxy as single point for directing URLs that require some sort of authentication.
- For ease of URL generation, configure ezproxy to make decisions regarding how to handle on and off campus traffic.
- Use ezproxy's SPUEdit directives to redirect traffic to shib-enabled resources through SessionInitiators provided by Resource Providers
- For resources that allow both IP and shib authentication, take advantage of SPUEdit IP directives (-ExcludeIP, -IncludeIP, -AutoLoginIP) to redirect on campus traffic to the original URL, so as to bypass forcing on campus users to log in.