IdPDiscovery

Identity Provider Discovery

When a user would like to login with Shibboleth after accessing a resource directly, the user's home IdP must be identified. That process is known as IdP discovery, and it can be done in a lot of different ways.

Most of the methods for performing IdP discovery involve asking the user directly, because the user can always answer the question best. The question can be posed in many different ways. Since discovery introduces a task for the user and thus creates user interface challenges, doing discovery right is an art that is paramount for the success of any Shibboleth deployment.

Where to Begin

Discovery can either be done using a flat, static page that relies on a fixed, known set of possible IdP's, or done using a dynamic discovery service, a separate app that can generate a set of options based on metadata, present those options to the user, and send the user to the selected home.

Flat Page Discovery

Static HTML leveraging the SP's SessionInitiator support is often sufficient for discovery when there is a limited and static set of IdP's to choose from, and is simple and clean to implement.

From a Known Home

If the user is coming from a source that knows the user's home IdP by definition, that source can send the user directly to the resource, either having already authenticated for that SP, or having preselected the IdP for the user. This can be done on campus portals, URL paths specific to users from one customer, or on library pages, for example.

There are two ways to send the user back to a known home IdP.

The best by far is to utilize a known SessionInitiator at the SP protecting the desired resource and supply it the encoded EntityID for that IdP. For example, TestShib had a SessionInitiator located at `https://sp.testshib.org/Shibboleth.sso/TestShib`. If supplied an appropriately encoded entityID parameter of `https://idp.testshib.org/idp/shibboleth`, the SP will examine metadata to determine that it recognizes the IdP, and then select the right endpoints at the IdP and SP automatically. It will dispatch the AuthnRequest directly to the IdP, and the user, upon authentication, will arrive at the resource immediately. The complete request will then look like:

```
https://sp.testshib.org/Shibboleth.sso/TestShib?entityID=https%3A%2F%2Fidp.testshib.org%2Fidp%2Fshibboleth
```

This can be placed in any link on any page. The user will authenticate if they haven't already, and they will end up at the application. A separate `target` can also be added to specify the page at which the user should land.

If the SessionInitiator is not known, then it's possible to forge an AuthnRequest on behalf of the SP based only on information that is present in metadata. This was a common approach with the query string-based authentication requests in Shibboleth 1.x, but it's considerably more difficult with the XML-based, encoded requests in SAML 2.0. It's listed here as an option, but strongly recommended against.

At the Application

If the application knows the set of IdP's that might access it and the set is exceedingly small, then a simple approach to discovery on a flat page might work. For example, TestShib's registration application supports login only through OpenIdP and ProtectNetwork. The simplest and successful approach was to imbed logos for each of the providers on the page in the application itself, with the links going to a SessionInitiator for each provider. These links are very similar to the links described above, but they use a special SessionInitiator for each IdP, and they specify a `target` resource for the user to land at after authentication.

```
<a href="https://www.testshib.org/Shibboleth.sso/ProtectNetwork?target=https%3A%2F%2Fwww.testshib.org%
2Ftestshib-two%2Fauth-pages%2Fauth.jsp">
   <img src="images/PN_sign-in.gif" alt="Login with ProtectNetwork" border="0">
</a>

<a href="https://www.testshib.org/Shibboleth.sso/OpenIdP.org?target=https%3A%2F%2Fwww.testshib.org%2Ftestshib-
two%2Fauth-pages%2Fauth.jsp">
   <img src="images/openidp.png" alt="Login with OpenIdP.org" border="0">
</a>
```

```
<center>
<table cellpadding="5" border="1"><tr><td width="200" height="250"><table><tr><td width="200" height="125"><h3
style="text-align: center; margin-top: 20px">
        <h3 style="text-align: center; margin-top: 20px">
                <a href="https://www.testshib.org/Shibboleth.sso/ProtectNetwork?target=https%3A%2F%2Fwww.
testshib.org%2Ftestshib-two%2Fauth-pages%2Fauth.jsp">
                <img src="http://www.testshib.org/testshib-two/images/PN_sign-in.gif" alt="Login with
ProtectNetwork" border="0"></a></h3></td></tr><tr><td height="125">
                <h3 style="text-align: center; margin-top: 20px"><a href="https://idp.protectnetwork.org
/protectnetwork-idp/registration.html" target="_new">
                <img src="http://www.testshib.org/testshib-two/images/PN_register.gif" alt="Register a new
ProtectNetwork identity" border="0"></a><br/><br/>
        </h3></td></tr></table>
        </td><td width="175" height="250"><h3 style="text-align: center; margin-top: 20px">
        <a href="https://www.testshib.org/Shibboleth.sso/OpenIdP.org?target=https%3A%2F%2Fwww.testshib.org%
2Ftestshib-two%2Fauth-pages%2Fauth.jsp">
                <img src="http://www.testshib.org/testshib-two/images/openidp.png" alt="Login with OpenIdP.org"
border="0"></td></tr>
</table>
</center>
```

Discovery Service

An IdP Discovery Service (DS) is a service that presents a standard interface for users to select their IdP from. A DS presents in some form, highly customizable, a set of IdP's from which the user can choose. After the user makes a selection, the DS redirects the user to the SP, which then formulates the AuthnRequest based on the user's selection.

The user is generally sent directly to the DS in a redirect, but more modern DS implementations (e.g. SWITCH's WAYF, the Embedded Discovery Service or CESNET's wayf) also allow an application to directly embed a discovery interface into a page. The interface can be customized by that page.

The DS can be operated by the resource, or it can be run as a central, shared service. There are advantages and drawbacks to either approach.

Run with the Resource

If the resource operates its own DS, it can present the smallest list of potential choices to the user. The resource knows the full set of IdP's it will accept, and it knows which federations it has partnerships with. This is particularly valuable for services serving a large number of communities, or a small subset of a large community.

It can also hint based on additional knowledge it might have, such as a table matching the user's IP address to likely home organizations. The resource can also integrate the branding, look, and feel of the discovery service into its own site.

As a drawback, this requires the SP to maintain its own DS, and can lead to an inconsistent experience for users. Please consult the DiscoveryService documentation to understand how to set up and operate a DS.

Run Centrally

In user testing, consistency in discovery experience has been repeatedly rated as the most important feature. Operation of a central DS creates that consistent user experience. Also, because the DS is centralized, selections made there can persist across all applications that use the shared DS. This can dramatically reduce the number of times the user must select a home.

The central DS is usually operated by a federation. This creates problems for applications that serve multiple federations. As the prospects for a global discovery service appear limited for both user interface and practical reasons, this drawback is likely to linger.
A central DS not operated by one federation, that also allows for integration methods with a user experience similar to a DS "run with the resource", is SeamlessAccess.

Interface Approaches

Buttons

Drag-down menus

Auto-completion

Text Boxes

Infocard