AddSeparateApplication

Create a separate Application/providerId on the same webserver

(Shibboleth SP v1.3)

You usually don't want to release all attributes to every application that's running on a particular webserver. Though this is of less importance if one person manages all applications on that server, it becomes extremely important on a shared webserver! To control the release of attributes at the IdP side, you need the SP to identify every application with a separate providerId. This can all be configured in shibboleth.xml (and appropriate adjustments at the IdP) as explained on this page.

However, each application SHOULD live on a separate virtual host from the others. Failing to do this will not provide any real separation of applications because any resource in one application can generally trick the client into supplying the cookie issued by another.

At the SP

Map the separate application to a different applicationId (RequestMap element)

The `<RequestMap>` element contains mappings from request-URLs to `applicationIds`. Take a look at this piece of configuration in the default `shibboleth.xml`:

```
<RequestMapProvider type="edu.internet2.middleware.shibboleth.sp.provider.NativeRequestMapProvider">

    <RequestMap applicationId="default">

        <!--
        This requires a session for documents in /secure on the containing host with http and
        https on the default ports. Note that the name and port in the <Host> elements MUST match
        Apache's ServerName and Port directives or the IIS Site name in the <ISAPI> element below.
        -->
        <Host name="sp.example.org">
            <Path name="secure" authType="shibboleth" requireSession="true" exportAssertion="true"/>
        </Host>

        <!-- this line below: -->
        <Host name="admin.example.org" applicationId="foo-admin" />
    </RequestMap>

</RequestMapProvider>
```

The `<RequestMap>` element always has an `applicationId` attribute with a **fixed** value of default that matches another configuration block a bit lower in shibboleth.xml.

Note the line that contains `applicationId="foo-admin"`. This is a mapping of the request-URL to a separate `applicationId`. This is absolutely necessary to create a separate application since applications are defined by `applicationIds`. You can define a separate applicationId for any `<Host>` or `<Path>` element.

Important Note: for this security-contraint that is defined in the Shibboleth configuration to work, the request needs to pass through the Shibboleth module /filter. For IIS this means that the ISAPI filter should be enabled on this location (that's usually ok). For Apache httpd this means that you need to enable the module on this location in the webserver configuration or a .htaccess file (this is usually not the case). For Apache httpd, you can use these directives to enable the modules in "lazy session" mode so it processes each request:

```
<Location />
    AuthType shibboleth
    require shibboleth
</Location>
```

Adjust this configuration to fit your needs.

Make the new applicationId identify itself with a separate providerId (Applications element)

The next step is to map the new, separate `applicationId` to a new, separate `providerId`. This is also done in `shibboleth.xml`. Take a look at this piece of configuration:

```
<Applications id="default"
  providerId="https://sp.example.org/shibboleth"
  homeURL="https://sp.example.org/"
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">

   <!-- ... -->

    <Application id="foo-admin" providerId="https://admin.example.org">

        <!--
        ### this is *necessary* if you are subsetting a <host>,
        use the default /Shibboleth.sso when configuring for vhosts
        -->

        <!--
        <Sessions lifetime="7200" timeout="3600" checkAddress="false"
          handlerURL="/secure/admin/Shibboleth.sso" handlerSSL="true"
          cookieProps="; path=/secure/admin; secure">
        </Sessions>
        -->

    </Application>

    <!-- ... -->

</Applications>
```

Each separate application also needs a separate Shibboleth handler. That's what the `<Sessions>` elements takes care of. (When using a version prior to 1.3b you **MUST** specify an `AssertionConsumerService Location`. Forgetting this will result to the following error: "Shibboleth handler invoked at an unconfigured location".) When using vhosts, you will notice that `Shibboleth.sso` is already listening on all vhosts so there's no need for another `Sessions` element if the default's properties are OK.

Also take a look at the `cookieProps`, they make sure that THIS Shibboleth session stays within THIS application (`path` to restrict access to the cookie within this path; `secure` option to restrict transport of the cookie only when using https aka ssl-secured).

Note that this separate application can be configured to only see the `EduPersonPrincipalName` attribute of the user. If you use this configuration where you specify certain attributes to be realeased, then you will have to specify EVERY attribute that you want to release to this application. If you don't specify any attributes, Shibboleth queries for all attributes as defined in the parent element.

The SP is ready now.

At the IdP

It is obvious that the new application will also identify itself to the IdP with its new,separate `providerId`. That was actually the point of this entire howto. So the modifications will require some changes at the IdP too.

metadata

Using Shib v1.3, you will have to create separate metadata for each application since the Shibboleth handler. Copy the the entire <EntityDescriptor> element of the providerId of the "default" `applicationId` . The only changes necessary are the `entityID` and the `AssertionConsumerService Location` values. This can be done because you are using the same credentials as the default application.

ARP

The whole point was to create separate rules at the IdP to release attributes. So you should now adjust your `arp.sites.xml` to match your needs.

You have now separated an application from the default application in a simple way.