Setting up a Shibboleth SP

Overview

This document explains how to set up a Shibboleth Service Provider (SP) and integrate it with MyAccess. Even though this is for setting up with MyAccess, the steps are the same for integrating with any Identity Provider (IdP).

This document only covers installing a SP on CentOS (which is similar to RHEL) machine, i.e., it does not cover setting up a SP on Windows. With that said, parts of the configuration of the SP should apply equally to Windows as well as other platforms. It also assumes that you are using Apache 2.2.x.

Install the SP (shibd)

This document is about installing `shibd`, the C program which runs as its own service on the server. In this case, `shibd` **is** Shibboleth, i.e., the "Service Provider".

Install with yum

It is recommended that you install the SP via yum, as it will be easy to update the software when there are any security fixes, upgrades, etc.

Before you can use yum, you need to download a .repo file containing the info about where Shibboleth is located. Get the .repo file for your OS here:

http://download.opensuse.org/repositories/security://shibboleth/

And place the file in:

```
/etc/yum.repos.d/
```

Once you have done that, you can install shibd.

For 32-bit:

```
sudo yum -y install shibboleth
```

For 64-bit:

```
sudo yum -y install shibboleth.x86_64
```

Install manually

If the above does not work (it did not work for me because yum was looking for versions of files that did not exist in the repo), then you can install it manually.

> **Note**
>
> The remaining install instructions are assuming 64bit install on CentOS 5. If this is not your platform, be sure to adjust the commands as appropriate.

If installing manually, you need to install the uxixODBC devel package as log4shib requires this package be installed first:

```
sudo yum -y install unixODBC-devel.x86_64
```

Next, download the RPMs into a directory (I recommend that you create a directory just for these RPMs as it will make the install command easy):

```
wget http://download.opensuse.org/repositories/security://shibboleth/CentOS_5/x86_64/log4shib-1.0.4-1.3.x86_64.
rpm \
     http://download.opensuse.org/repositories/security://shibboleth/CentOS_5/x86_64/xerces-c-3.0.1-6.3.x86_64.
rpm \
     http://download.opensuse.org/repositories/security://shibboleth/CentOS_5/x86_64/xml-security-c-1.5.1-4.3.
x86_64.rpm \
     http://download.opensuse.org/repositories/security://shibboleth/CentOS_5/x86_64/xmltooling-1.3.3-1.2.
x86_64.rpm \
     http://download.opensuse.org/repositories/security://shibboleth/CentOS_5/x86_64/opensaml-2.3-1.9.x86_64.
rpm \
     http://download.opensuse.org/repositories/security://shibboleth/CentOS_5/x86_64/shibboleth-2.3.1-1.3.
x86_64.rpm
```

Once the files have been downloaded, install them as follows:

```
sudo rpm -ivh *.rpm
```

Now, you can verify that shibboleth has been installed:

```
sudo yum info shibboleth
```

and you should get output that looks something like this:

```
Installed Packages
Name       : shibboleth
Arch       : x86_64
Version    : 2.3.1
Release    : 1.3
Size       : 5.5 M
Repo       : installed
Summary    : Open source system for attribute-based Web SSO
URL        : http://shibboleth.internet2.edu/
License    : Apache 2.0
Description: Shibboleth is a Web Single Sign-On implementations based on OpenSAML
           : that supports multiple protocols, federated identity, and the extensible
           : exchange of rich attributes subject to privacy controls.
           :
           : This package contains the Shibboleth Service Provider runtime libraries
           : and Apache module(s).
```

Make Sure shibd Runs at Startup

Now, to make sure `shibd` runs at system startup time, type the following commands (check with your local sys admins fist to be sure this is OK with them):

```
sudo /sbin/chkconfig --add shibd
sudo /sbin/chkconfig --level 235 shibd on
```

To make sure `shibd` is configured properly, type the following:

```
sudo /sbin/chkconfig --list shibd
```

The output should look like this:

```
shibd                0:off       1:off       2:on        3:on        4:off       5:on        6:off
```

Configuration

Important notes about configuration can be read here:

https://spaces.internet2.edu/display/SHIB2/NativeSPLinuxRPMInstall

Once that is done, then come back to this section for further configuration information.

The install process places all Shibboleth-related configuration files into `/etc/shibboleth`. The files in this directory that you will be editing are:

- attribute-map.xml
- shibboleth2.xml

Configure shib.conf

Edit `/etc/httpd/conf.d/shib.conf` to protect resources by Shibboleth. For instance, the Library manages the CLE, which only has one page that needs to be protected (as application-level security takes care of regular session management), so the config for this application is:

```
<Location /auth/shibboleth/index.php>
  AuthType shibboleth
  ShibRequestSetting requireSession 1
  require valid-user
</Location>
```

Now, when a user tries to go to the URI `/auth/shibboleth/index.php` with their browser, Shibboleth (`shibd`) will intercept it and make sure the user is authenticated.

Note

All files ending in `.conf` which are located in the `conf.d` directory are automatically added to the Apache config when Apache is started. If you are not using the standard CentOS-installed Apache, then you can configure /etc/shibboleth/apache22.config, and then include this file into your Apache config by adding the following to the bottom of your httpd.conf file:

```
Include /etc/shibboleth/apache22.config
```

Configure httpd.conf

In order for Shibboleth to work properly, Apache has to have the proper hostname configured. So, either in the main `httpd.conf` or in the vhost section, make sure `ServerName` is the actual hostname of the service, **without** the port number. So, if the service name is host.subdomain.ucsf.edu, then the `ServerName` line should look like this:

```
ServerName host.subdomain.ucsf.edu
```

Shibboleth also requires that `UseCanonicalName` be set to on, so that line in `httpd.conf` should look look like this:

```
UseCanonicalName On
```

Once the Apache changes have been made, do an Apache `configtest` to make sure all is well, and if so, restart Apache.

Configure attribute-map.xml

MyAccess (and other IdPs) will send your application attributes which you ask them to send (this is a process where you ask the MyAccess personelle to configure their IdP to do this for you, i.e., there is no programatic way to get back attributes that your SP is not authorized to get back, so you have to ask for them and they have to configure the IdP to pass them to your SP).

Again, using the CLE as an example, we get back the following attributes from MyAccess:

- eduPersonPrincipalName (eppn)
- cn
- givenName
- sn (surname)
- mail
- employeeNumber
- ucsfEduIDNumber

In `attribute-map.xml`, uncomment the following lines:

```
    <Attribute name="urn:mace:dir:attribute-def:cn" id="cn"/>
    <Attribute name="urn:mace:dir:attribute-def:sn" id="sn"/>
    <Attribute name="urn:mace:dir:attribute-def:givenName" id="givenName"/>
    <Attribute name="urn:mace:dir:attribute-def:mail" id="mail"/>
```

Below those attributes, add the following attributes:

```
    <Attribute name="urn:mace:dir:attribute-def:employeeNumber1" id="employeeNumber1"/>
    <Attribute name="urn:mace:dir:attribute-def:ucsfEduIdNumber" id="ucsfEduIdNumber"/>
```

Since this is an XML document, comments are made with <!~~--~~ and ~~--~~> marks, so be sure to close any comment that you open.

Next, uncomment the OID versions of these attributes:

```
    <Attribute name="urn:oid:2.5.4.3" id="cn"/>
    <Attribute name="urn:oid:2.5.4.4" id="sn"/>
    <Attribute name="urn:oid:2.5.4.42" id="givenName"/>
    <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
```

And then add the following below those attributes:

```
    <Attribute name="urn:oid:2.16.840.1.113730.3.1.3" id="employeeNumber1"/>
    <Attribute name="urn:oid:1.3.6.1.4.1.20319.1.1.1.1" id="ucsfEduIdNumber"/>
```

If you need other attributes, then you will need to consult with MyAccess for the names of the attributes, and the OIDs used for them.

> **Note**
>
> `eduPersonPrincipalName` (or `eppn`) is configured by default, as this is the attribute which most IdPs assert, by default, about an individual. So you do not need to uncomment it.

Configure shibboleth2.xml

`shibboleth2.xml` is where most of the configuration goes, and this is where everything will start to make sense.

ApplicationDefaults configuration

Next, find the line that begins with:

```
<ApplicationDefaults entityID="https://sp.example.org/shibboleth"
                        REMOTE_USER="eppn persistent-id targeted-id">
```

This needs to be edited with the `entityID` of your SP, which will be in the form:

```
https://hostname.ucsf.edu/shibboleth
```

You should be using the same hostname as you used above. And yes, you need to be using SSL!

You will also note the attribute `REMOTE_USER`, which is the HTTP `REMOTE_USER` header variable. The `shibd` daemon populates the `REMOTE_USER` header with the value of `eppn` (`eduPersonPrincipalName`).

> **Note**
>
> At UCSF, the value for `eduPersonPrincipalName` is made up using part of a person's UCSF ID Number, with `@ucsf.edu` appended. For example, if your ID Number is 021234567, then your `eduPersonPrincipalName` will be 123456@ucsf.edu.

In this same section, you want to edit the SSO attribute for the MyAccess (test) IdP.

Find the section that looks like this:

```
<SSO entityID="https://idp.example.org/shibboleth"
     discoveryProtocol="SAMLDS" discoveryURL="https://ds.example.org/DS/WAYF">
  SAML2 SAML1
</SSO>
```

Change it to look like this:

```
<SSO entityID="https://d5n1.ucsf.edu/idp/shibboleth"
     discoveryProtocol="SAMLDS" discoveryURL="https://ds.example.org/DS/WAYF">
  SAML2 SAML1
</SSO>
```

> **Note**
>
> Of course, change `d5n1.ucsf.edu` to `dp.ucsf.edu` if you are integrating with production.
>
> Also, if you are using a discovery service, just like the note states in the XML above the SSO attribute, remove the value for `entityID`, and put the URL of your discovery service as the value for `discoveryURL`

Metadata configuration

Ask MyAccess for a copy of their test IdP Metadata file. Once you get it, put it in `/etc/shibbolth`, so something like this:

```
/etc/shibboleth/myaccess-test-metadata.xml
```

Now, back in the `/etc/shibboleth/shibboleth2.xml` file, find the section that begins with "Example of locally maintained metadata.", as this is where you will be putting information about the MyAccess metadata (for their test and or production IdP), and the InCommon metadata (if you need to use the InCommon metadata).

Add an entry for the MyAccess metadata file:

```
<MetadataProvider type="XML" file="/etc/shibboleth/myaccess-test-metadata.xml"/>
```

> **Note**
>
> If you need this SP to interact with an IdP that is part of InCommon (like the production MyAccess IdP), then add a provider for the InCommon metadata file. Since this is detailed on the InCommon section of the Internet2 wiki, I will not repeat it here:
>
> https://spaces.internet2.edu/display/InCCollaborate/Configuration+Guide#ConfigurationGuide-ConsumingInCommonMetadata

Start shibd

Now that all of this is configured, you can start the `shibd` daemon. On Linux you do this using the following command:

```
sudo /sbin/service shibd start
```

`shibd` logs things to the following two files:

```
/var/log/shibboleth/shibd.log
/var/log/shibboleth/transaction.log
```

If `shibd` does not start, then you will want to tail the `shibd.log` file to see where the error is located. Most likely it will be because of an XML error (like forgetting to close a comment).

Your SP Metadata

Just like an IdP, your SP also has its own metadata, and in order for IdPs to be able to trust your SP and sign information specifically for the SP, it needs the SPs metadata.

> **Note**
>
> If this SP is going to allow users from other institutions, i.e., users who are not using MyAccess for authentication, then you need to request that MyAccess register your SP with InCommon. For production this is a must. For instance, if you want people for LBNL to use your SP, then MyAccess needs to register your SP with InCommon so that the LBNL IdP gets your metadata when its metadata file refreshes.

You can access your SP metadata by going to this URL:

```
https://hostname.ucsf.edu/Shibboleth.sso/Metadata
```

Of course, replace `hostname` with the hostname of your server.

If you can successfully download the metadata, then you know `shibd` is running correctly.

Logout

Shibboleth supports logout from the SP and the ability to be sent to a location after logout, like an IdP logout page if the IdP supports logout. The MyAccess IdP has a logout page which will end the IdP (SSO) session for the user at the IdP itself. In other words, if a browser is redirected to this URL the browser will be forced to log into Shibboleth the next time the user tries to log into a Shibboleth-protected service **and** the user does not already have a session established with that service. So, existing sessions at other SPs are still valid, but any new session would require authentication.

Edit MyAccess Metadata

To enable logout for the MyAccess IdP, add the following the MyAccess IdP metadata (in the IDPSSODescriptor section):

```
<SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
    Location="https://d5n1.ucsf.edu/idp/shib_logout.jsp"/>
```

Also, if you want the MyAccess IdP logout page to display a URL which the user can use to return to the application, you can add a `url=` param to the end of `shib_logout.jsp`. So, if we want to return to the wiki, we would do this:

```
<SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
    Location="https://d5n1.ucsf.edu/idp/shib_logout.jsp?url=https://wiki-dev.library.ucsf.edu/"/>
```

Note

Change `d5n1.ucsf.edu` to `dp.ucsf.edu` if you are integrating with production.

If you are using more than one IdP for login to your application and each IdP supports logout, then if there is a `SingleLogoutService` present in the metadata for the IdP, the `shibd` process will send the user to the correct location when the browser is sent to `/Shibboleth.sso/Logout`.

Edit shibboleth2.xml

In order for the above logout mechanism to work for the IdP, the following must be configured in the `Sessions` section of `shibboleth2.xml`:

```
<LogoutInitiator type="Chaining" Location="/Logout" relayState="cookie">
    <LogoutInitiator type="SAML2" template="bindingTemplate.html"/>
    <LogoutInitiator type="Local"/>
</LogoutInitiator>
```

Integrate With MyAccess

At this point you are ready to integrate with MyAccess. You should open up a service ticket with ITS (http://help.ucsf.edu/ then click on "Submit a ticket for ITS or School of Nursing IT") and include the following information:

1. Subject indicating that the request is for "MyAccess Shibboleth [test or production]"
2. Attributes you want to get back from their IdP (and if you want ones that were not covered above, then you need to ask them for the OID for the attribute and configure it in `attribute-map.xml`)
3. URL for your metadata (so that they can download the metadata, or attach the metadata file to the ticket)

If you have not done so already, ask them to send you their test IdP metadata and place it in the location which we covered above.

Test Your SP

Once you have the above set up, you can test `shibd` without even having an application configured to use Shibboleth. Just create a simple form on your shibbolized webserver that looks like this:

```
<form method="post" action="/Shibboleth.sso/Login">
<input type="hidden" name="target" value="https://hostname.ucsf.edu/uri-you-protected-in-apache22.config" />
<input type="hidden" name="providerId" value="https://d5n1.ucsf.edu/idp/shibboleth" />
<input type="submit" class="save" value="Log in via MyAccess" />
</form>
```

Of course, replace `hostname` with the hostname of your server and `uri-you-protected-in-apache22.config` with the actual URI that you put into `apache22.config`. If you did not protect a URI in `apache22.config`, then do that now, and restart Apache.

Now, tail the following file:

```
tail -f /var/log/shibboleth/transaction.log
```

And then submit the form. If all goes well, you will be forced to authenticate against the MyAccess test IdP, and once that is successful, you will be back on your server, and the `transaction.log` file should have something like this in it:

```
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]: New session (ID: _<some id>) with (applicationId:
default) for principal from (IdP: https://d5n1.ucsf.edu/idp/shibboleth) at (ClientAddress: 128.218.15.62) with
(NameIdentifier: _<some id>) using (Protocol: urn:oasis:names:tc:SAML:2.0:protocol) from (AssertionID: _<some
id>)
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]: Cached the following attributes with session (ID:
_991c886fbda8119389f5435674272987) for (applicationId: default) {
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]:         eppn (1 values)
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]:         cn (1 values)
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]:         sn (1 values)
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]:         givenName (1 values)
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]:         employeeNumber1 (1 values)
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]:         mail (1 values)
2010-08-23 14:47:12 INFO Shibboleth-TRANSACTION [80]: }
```

If you do not see the this, then there is something wrong with the configuration. Check the `shibd.log` file for more information.

Resources

- Internet2 Shibboleth Installation Page
- SP Instructions for Linux
- SP Instructions for OS X