SP Shibboleth Installation – Linux (Ubuntu)

0. Foreword

We suppose that you're using a server version of a recent LTS version of Ubuntu Linux (based on 10.04 LTS for this document). We also suppose that you've an Apache web server running, giving access to a web application. This document will show you how to *shibbolize* a web application, turning it into a federated **Service Provider (SP)**.

For that purpose, we will suppose that our application is only a simple PHP dummy *index.php* file displaying *"Hello world"* when accessing the website. We will enhance it to integrate a Shibboleth authentication phase, and display *"Hello" [name] coming from [institution]"*, showing how to access simple attributes using PHP. Also, all the commands need to be entered with root permission, so it is probably a good idea to start in your terminal with : sudo su

1. Install Shibboleth SP daemon

You may install the Shibboleth daemon either by compiling the sources or simply by installing the deb package by issuing following : sudo apt-get install libapache2-mod-shib2 To install from sources, you may download the latest version of Shibboleth-SP here (www.shibboleth.net/downloads/service-provider/latest).
Untar the archive, configure, make and install it. You may follow the *README* file present in the archive.

2. Configure Shibboleth SP daemon

The configuration of the daemon resides in */etc/shibboleth/shibboleth2.xml*. Open it with a text editor and adjust following parts.

2.1. Define the entityID for your SP

<ApplicationDefaults id="default" policyId="default" entityID="https://helloworld.belnet.be/shibboleth" REMOTE_USER="eppn persistent-id targeted-id" signing="true" encryption="true">

You need to setup an *entityID* that will be used to distinguish your SP among the others in the Federation. *REMOTE_USER* contains the list of possible attributes that may be used to authenticate users connecting to your site.

2.2. Define the session parameters

It is a good deal to put *handlerSSL="true"* to force *HTTPS* protocol to be used. <Sessions lifetime="28800" timeout="3600" checkAddress="false" handlerURL="/Shibboleth.sso" handlerSSL="true" exportLocation="http://localhost/Shibboleth.sso/GetAssertion" exportACL="127.0.0.1" idpHistory="false" idpHistoryDays="9">

2.3. Define a session initiator

It will be used to handle the requests to your site and redirect them to either an *IdP* or a *Discovery (WAYF)* page.

<!-- SessionInitiators handle session requests and relay them to a Discovery page, or to an IdP if possible. Automatic session setup will use the default or first element (or requireSessionWith can specify a specific id to use). -->

<!-- Default example directs to a specific IdP's SSO service (favoring SAML 2 over Shib 1). -->

<SessionInitiator type="Chaining" Location="/Login" isDefault="true" id="Intranet" relayState="cookie" entityID="https://your-idp.yourdomain.be/idp/shibboleth" forceAuthn="true"> <SessionInitiator type="SAML2" acsIndex="1" template="bindingTemplate.html"/> <SessionInitiator type="Shib1" acsIndex="5"/> </SessionInitiator> <!-- An example using an old-style WAYF, which means Shib 1 only unless an entityID is provided. --> <SessionInitiator type="Chaining" Location="/WAYF" id="WAYF" relayState="cookie"> <SessionInitiator type="SAML2" acsIndex="1" template="bindingTemplate.html"/> <SessionInitiator type="Shib1" acsIndex="5"/> <SessionInitiator type="WAYF" acsIndex="5" URL="https://federation.belnet.be/WAYF"/> </SessionInitiator> <!-- An example supporting the new-style of discovery service. --> <SessionInitiator type="Chaining" Location="/DS" id="DS" relayState="cookie"> <SessionInitiator type="SAML2" acsIndex="1" template="bindingTemplate.html"/> <SessionInitiator type="Shib1" acsIndex="5"/> <SessionInitiator type="SAMLDS" URL="https://ds.example.org/DS/WAYF"/> </SessionInitiator>

2.4. Define the metadata sources

Adjust the following sections :

<!-- Chains together all your metadata sources. ---> <MetadataProvider type="Chaining"> <!-- Example of remotely supplied batch of signed metadata. --> <MetadataProvider type="XML" uri="https://federation.belnet.be/federation-metadata.xml" backingFilePath="/etc/shibboleth/federation-metadata.xml" reloadInterval="7200"> <MetadataFilter type="RequireValidUntil" maxValidityInterval="2419200"/> <MetadataFilter type="Signature" certificate="/etc/ssl/certs/certificate.federation.belnet.be.crt"/> </MetadataProvider> <!-- Example of locally maintained metadata. --> <!-- <MetadataProvider type="XML" file="/etc/shibboleth/idpstaff.belnet.be.xml"/> >!-- </MetadataProvider>

Of course, you'll need to provide your own SP metadata to the *Belnet Federation Metadata Manager*. Next section explains how to setup your metadata.

2.5. Define the SP metadata file

Of course to make your site working in the Shibboleth federationn, you need to have your own metadata describing your SP. The easyest way to generate it is to use a neat little application that comes with the Shibboleth SP package you've installed :

shib-metagen -c /etc/ssl/certs/yourcertificate.crt -h yourhost.yourdomain > /etc/shibboleth/yourSP-metadata.xml

This command will generate a XML text file containing all your need for your metadata. You may type the command without anything to see the arguments that may be passed to it. You need to give it the public certificate used by the cryptographic part of our Shibboleth thins (it is the *-c* arg) and the name of your SP (*-h* arg). By default it will generate an *entityID* based on the hostname and default parameters (but you may change it with the *-e* argument.

As it prints everything out to the default output, it is better to redirect everything to a file that will handle the metadata (here */etc/shibboleth/yourSP-metadata.xml*).

2.6. Define the SP attributes policy

Edit the file *attribute-policy.xml*. This file permits to establish rules concerning the received attributes. A nice stuff is to let people enter your site if they only have specific attributes define in their user's profiles.

2.7. Define the SP attributes mapping rules

Edit the file *attribute-map.xml*. This file permits to prepare the received attributes to be used by your web SP application.

3. Start Shibboleth SP daemon

Issue following command to start the Shibboleth SP daemon :
/etc/init.d/shibd restart

Don't forget that each time you proceed to changes to the config of your Shibboleth SP daemon, you will need to restart it.

4. Setup Apache 2 to enable Shibboleth

It is quite easy to turn on Shibboleth :

- Enable the Shibboleth's Apache module :
  a2enmod shib2
- Add the following necessary snippet to the configuration of your web site :
  <Location> AuthType shibboleth ShibRequireSession on Require valid-user </Location>

Don't forget to restart Apache !

/etc/init.d/apache2 restart

There are other configuration parameters you may setup, but we propose you to explore the Shibboleth wiki site (https://wiki.shibboleth.net/confluence/#all-updates) to find out every capabilities !

5. *Shibbolized* application example

We will use a simple PHP example file to demonstrate how to *shibbolize*. Our web application consist of a single *index.php* file that display a classical *"Hello world !"* on the webpage.

<html> <body> <?php echo "Hello world !"; ?> </body> </html>

Now, if you have configured the Shibboleth SP daemon, added the necessary lines in the Apache configuration and started the *shibd* daemon, you may add *PHP* code to get the received attributes and process them into your application... <html> <body> <?php echo "Hello world !"; // Do we have any variables defined in attribute map if (isset($_SERVER['eppn'])){ echo 'Hello '.$_SERVER['cn'].' from

'.$_SERVER['o'] ; } // or any attributes starting with Shib- else { echo 'You are unknown !'; } ?> </body>
</html>