Add-ons

Introduction

Add-ons for HUBzero are available here. Currently these consist of a couple projects that have not yet been fully integrated into the the HUBzero packaging and installation process.

Separate execution host setup

Intro

Our standard open source install sets up a web server and execution host on a single machine. For smaller hubs, this is an adequate setup, however on hubs with greater needs, often separate (and sometimes multiple) execution hosts are required for an installation. The following is a set of directions for setting up an execution host.

Note, these directions are not complete step by step directions, but more of a guideline for setting up an execution host. A user should have a very thorough understanding of hub architecture before attempting to setup an additional execution host.

Setup steps

1. Do standard debian OS install
2. Configure hostname and /etc/sources.list appropiately
3. Setup standard hub through the openldap step
4. Install openvz kernel
5. Install hubzero mw-service on execution host 'apt-get install -y hubzero-mw-service'
6. Run 'mkvztemplate amd64 wheezy diego'
7. run 'hzcms configure mw-service --enable'
8. configure /etc/nslcd.conf and restart. /etc/nslcd.conf will need the following modifications:
   URI - modified to point to the ldap on the web host
   binddn - set to the search user dn on the webserver (do a 'slapcat | grep search' to get the DN for the search user on the web server)
   bindpw - set to the value contained for the LDAP-SEARCHPW in the /etc/hubzero.secrets file on the web server
9. Install hubzero mw-client and configure on execution host
10. copy /etc/mw-client/maxwell.key.pub from web host to /root/.ssh/authorized keys file on execution host
11. On web server, add execution host to tools component
    login to webserver admin section (webserver/administrator)
    select components->tools
    on host tab click on + sign in upper right to add an execution host
    When you are returned to the list of hosts, you should see two, one for the web server, likely called localhost and the IP for your execution host
    Under the provisions section, click on pubnet, sessions, and workspace for the new execution host
    Under the provisions section, uncheck everything but fileserver for your web server
12. Setup nfs server on web server
    'apt-get install nfs-kernel-server'

edit /etc/exports to export /home and /apps, something similar to this:
/home executionhost.ip.address(rw,no_subtree_check)
/apps executionhost.ip.address(rw,no_subtree_check)

13. Setup nfs client on execution host
apt-get install nfs-common
mount -t nfs webserver:/home /home
mount -t nfs webserver:/apps /apps
NOTE: user will want to add appropiate sections in the /etc/fstab file to remount these locations after a reboot. Something similar to:
your.webserver.org:/apps /apps nfs vers=3,rw 0 0
your.webserver.org:/home /home nfs vers=3,rw 0 0

InCommon

These instructions are for Debian 7 (wheezy) ONLY. They have not been updated for Debian 8

Introduction

This plugin provides some code necessary to allow your hub to accept credentials using the Shibboleth system. Most commonly, this implies membership in the InCommon network.

Shibboleth has some particular architectural demands, namely that it will install a new daemon and a new Apache module on your system. InCommon has some administrative demands, in that you will need to negotiate to get your hub added to their XML manifest as a service provider.

Installation

Shibboleth wiki entry on service provider installation

Debian

It is necessary on a Debian 7 Wheezy host to add the backports repository to install this the required packages, see https://backports.debian.org/Instructions/#index2h2

```
deb http://ftp.debian.org/debian wheezy-backports main
```

Then run the follow command to install and do some initial configuration for you such as enabling the module.:

```
apt-get -t wheezy-backports install libapache2-mod-shib2
```

```
aptitude install libapache2-mod-shib2
```

If your distribution does not have this package, refer to the Shibboleth wiki page above for information about other installation methods.

Redhat Enterprise Linux & other distributions

See the wiki page above for information on how to add the Shibboleth software to your list of repositories so that it can be installed and upgraded through yum, or, failing that, how to install from SRPMS.

Configuration

Shibboleth

Certificates

As root, run the script shib-keygen, which was installed as part of the package. This will generate a key pair for your service provider to use. No further configuration is required for this; the software will find the keys when the shibd service is restarted.

output

```
Generating a 2048 bit RSA private key
.....................................................................
.................+++
.....................+++
writing new private key to '/etc/shibboleth/sp-key.pem'
-----
```

/etc/shibboleth/attribute-map.xml

This file controls which attributes (bits of user information) the software will extract during login [when the identity provider makes them available](#).

Make sure the following pertinent attributes are not commented out in both forms of the “name” attribute.

eppn (username, probably already enabled in the shipped configuration):

```
    <Attribute name="urn:mace:dir:attribute-
def:eduPersonPrincipalName" id="eppn">
```

```
        <AttributeDecoder xsi:type="ScopedAttributeDecoder"/>
    </Attribute>
    <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6" id="eppn">
        <AttributeDecoder xsi:type="ScopedAttributeDecoder"/>
    </Attribute>
```

Name & email (probably not enabled by default):

```
    <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:displayName" id="displayName"/>
    <Attribute name="urn:mace:dir:attribute-def:mail" id="mail"/>
    <Attribute name="urn:oid:2.5.4.4" id="sn"/>
    <Attribute name="urn:oid:2.5.4.42" id="givenName"/>
    <Attribute name="urn:oid:2.16.840.1.113730.3.1.241" id="displayNam
e"/>
    <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
```

/etc/shibboleth/shibboleth2.xml

This is the main configuration, which controls how the software federates with identity providers.

First, replace $YOUR_HOSTNAME with, uh, your hostname, in the entityID attribute near the top of the file:

```
       <ApplicationDefaults entityID="https://$YOUR_HOSTNAME/login/shi
bboleth" REMOTE_USER="eppn persistent-id targeted-id">
```

In the block, delete or comment-out any SSO or SessionInitiator blocks that shipped, and add the two listed below, again interpolating your real hostname. This tells the software to check with the CMS plugin about where to redirect for a given authentication request, and allows the CMS to selectively enable providers.

```
        <Sessions lifetime="28800" timeout="3600" relayState="ss:mem"
                  checkAddress="true" handlerSSL="true" cookieProps="h
```

```
ttps">
            <SSO discoveryProtocol="SAMLDS" ECP="true" discoveryURL="h
ttps://$YOUR_HOSTNAME/login?authenticator=shibboleth&wayf">
                  SAML2 SAML1
            </SSO>
            <SessionInitiator type="Chaining" Location="/login/shibbol
eth" isDefault="true" id="Login">
                <SessionInitiator type="SAML2" template="bindingTempla
te.html"/>
                <SessionInitiator type="Shib1"/>
                <SessionInitiator type="SAMLDS" URL="https://$YOUR_HOS
TNAME/login?authenticator=shibboleth&wayf"/>
            </SessionInitiator>
            <!-- Default <Handler> tags not pictured, but they should
stay -->
       </Sessions>
```

If you run into issues where you seem to be stuck in a redirect loop between the idp and the sp, changing the cookie properties to use a less specific path may help.

```
cookieProps="; path=/; secure; HttpOnly"
```

If this is a production machine you will want to set a real email for the support contact:

```
       <Errors supportContact="support@$YOUR_HOSTNAME"
            helpLocation="/about.html"
            styleSheet="/shibboleth-sp/main.css"/>
```

Finally, you will need to configure how and where the software looks for metadata about identity providers. This is just a list of providers you can support, including some helpful annotations like where the service URLs and what public key to use when communicating with it.

Metadata provider: TestShib

For development and test machines it is often useful to use TestShib, and its configuration looks like this, below the Sessions tag and at the same scope:

```
      <MetadataProvider type="XML" uri="http://www.testshib.org/metada
ta/testshib-providers.xml" backingFilePath="testshib-two-idp-
metadata.xml" reloadInterval="180000"/>
```

Visit the TestShib site for more information about how to set this up, if you're interested. Hopefully you do not need the "Install" selection, but pick up from "Register". During "Configure" it recommends replacing your whole shibboleth2.xml with one it generated. Make a backup if you do, or else just add the MetadataProvider above to your existing configuration.

When you reach "Test", see below for the CMS configuration that will add TestShib to the list of available identity providers on your hub.

Metadata provider: InCommon

If your plans include membership in the InCommon consortium, this is the incantation, below the Sessions tag and at the same scope:

```
        <MetadataProvider type="XML" uri="https://wayf.incommonfederat
ion.org/InCommon/InCommon-metadata.xml" backingFilePath="federation-
metadata.xml" reloadInterval="7200">
            <MetadataFilter type="RequireValidUntil" maxValidityInterv
al="2419200"/>
            <MetadataFilter type="Signature" certificate="inc-md-
cert.pem"/>
        </MetadataProvider>
```

Install https://ds.incommon.org/certs/inc-md-cert.pem as /etc/shibboleth/inc-md-cert.pem so it's available for this provider.

Metadata provider: others?

If you are doing one-on-one negotiations with identity providers the metadata situation gets a bit more hairy, but the identity providers in question will probably be able to guide your configuration.

Apache

Quoth the Shibboleth wiki entry on service provider installation:

- UseCanonicalName On
- Ensure that the ServerName directive is properly set, and that Apache is being started with SSL enabled.

Make sure installing the software enabled both the module shib2 and the support daemon shibd.

Typically this means that there is a symlink /etc/apache2/mods-enabled/shib2.load that points to /etc/apache2/mods-available/shib2.load and that this report works:

```
# service shibd status
[ ok ] shibd is running.
```

/etc/apache2/sites-enabled/{your-ssl-enabled-config-file}

Your EntityID is something like https://hostname/login/shibboleth, but the actual URL to pick up the login process again in CMS terms is more complicated, so we rewrite it. I recommend putting this statement as high as possible in the config (after RewriteEngine on) so that the “L“ast last triggers and you can be assured the URL is not subsequently rewritten by anything else you’re doing.

```
        RewriteCond     %{REQUEST_URI}          ^/login/shibboleth
        RewriteRule     (.*) /index.php?option=com_users&authenticator
=shibboleth&task=user.login [NC,L]
```

Bind an endpoint to the module. This is used during the login process and is also useful to get a basis for your service provider’s metadata, which is served at /Shibboleth.sso/Metadata when the request comes from localhost.

```
        <Location /Shibboleth.sso>
                SetHandler shib
        </Location>
```

You probably have a rule that directs all requests that appear to be for CMS content to the index.php bootstrap, and we need to note that /Shibboleth.sso isn’t CMS business, so make

sure you have a RewriteCond like this:

```
        RewriteCond     %{REQUEST_URI}          !^/Shibboleth.sso/.*$
[NC]
        RewriteRule     (.*)                    index.php
```

Finally, we actually protect the entityID location /login/shibboleth. We can redirect a user to this path to require them to make a Shibboleth login. Shibboleth won't know specifically how to do that so it will make a request to the wayf location defined above in shibboleth2.xml. This is part of the CMS that knows already which provider the user selected from the login page, so it spits back the appropriate identity provider entityId. From there the metadata is referenced to find the endpoint associated with that institution, and the user is sent to the login page. They come back to /login/shibboleth upon submission, but now the requirement to have a Shibboleth session is satisified, and the rewritten URL referencing user.login is served to complete the process.

```
        <Location /login/shibboleth>
                AuthType shibboleth
                ShibRequestSetting requireSession 1
                Require valid-user
        </Location>
```

Restart the shibd and apache2 services when satisified with this configuration.

CMS

Plugin

Log in to /administrator, choose Extensions and then Plugin Manager, and locate the Shibboleth plugin in the Authentication category.

If you would like to selectively hide the Shibboleth login options for testing, enter something in the "Testing mode key" field. This term must appear in the query string for the controls of the plugin to render. For example, if you enter "incommon" you should test the login page by visiting "/login?reset=1&incommon" (reset=1 in case it remembers your logging in with a different method, in which case you'll only see the controls for that method anyway).

The links section is useful only for testing, but it can be used to destroy a link between your test

account and a particular institution so that you can try it again.

The Institutions section is where you manage which options are presented on the login page.

An example of an entry here, for the TestShib public identity provider test mechanism described above:

```
Entity ID: https://idp.testshib.org/idp/shibboleth
Label: TestShib
Host: testshib.org
```

The entity ID must strictly match what you have in your metadata provider, but the label is free-form and the host is optional. The login page attempts to do a reverse-DNS of the user's IP to see if they are on a particular network. If it turned out in this case that the client was from *.testshib.org this option would be pre-selected in the plugin's controls.

Save your settings with the button near the top right of the page when you're done.

Solr-powered Search

Introduction

Apache Solr is a search engine platform which is relatively mature and has a lot of powerful and flexible configurations. There has been extensive work to implement it into the HUBzero CMS and is currently a work-in-progress.

Solr is an open-source, mature, and stable searching service that is built upon the Apache Lucene search engine. The service provides features which lend itself to scaling and has a rich open source community. It is a Java-based service which provides search results through HTTP. Many companies such as Instagram, eBay, and StubHub rely on Solr to provide advanced searching capabilities.

The integration with Solr is currently under heavy development. It is **strongly recommended** to test on a QA / Stage host before using in a production environment.

Installation & First Time Configuration

Step 1: Install the hubzero-solr package

A system administrator must install the hubzero-solr RedHat or Debian Package using a package manager such as yum or aptitude. The package contains a version of Apache Solr and the configuration necessary for Solr to integrate with the CMS.

```
For RedHat / CentOS:
$    sudo yum install hubzero-solr
```

```
For Debian:
$    sudo apt-get install hubzero-solr
```

Once installed the service will need to be enabled.

```
$    sudo service hubzero-solr start
```

Step 2: Configure Search Service in the CMS

The HUBzero CMS needs to know to use Solr Search instead of Basic Search. To do this, a Hub administrator will need to log into the Administrative Backend and Configure the Search Component.

You will need to set **Engine** to *Apache Solr.* Then click the "Solr tab".

The Solr tab's default settings will work for the open-source distribution.

HUBzero-hosted hubs are configured with different ports! The following scheme is used:

```
Development (dev.hub.org): 2090
Stage (stage.hub.org): 2091
Scan / QA (qa.hub.org): 2092
Production (hub.org): 2093
```

Click "Save and Close" to save the settings. If the hubzero-solr service is started and the correct settings were set in the steps above, the status screen should indicate that the search engine is responding.

If there were any issues with configuration, the following screen will appear.

This would be a point where a support ticket is filed for the system administrator to confirm that the service is running. Please include all configuration parameters contained in Step #3 when filing the ticket.

Step 3: Enable the Search Background Worker

In order to keep the search index fresh, a background worker is implemented to process data from the CMS and push it into the Solr service.

Currently the background worker is implemented as a Cron task that is called once a minute. There is work being done to develop a daemon which listens to CMS events and processes data without relying on Cron.

To setup the Cron-based worker a Hub administrator must go into the Administrative Backend, go to Components, Cron, and add the Task as shown below:

Click "Save and Close".

Step 4: Build the Initial Index

This implementation of Solr has hooks into the CMS which updates the index when a new record is added or marked for deletion. It will be necessary to add items which have been added before Solr was activated.

This operation should only need to be completed once. You will be unable to start this operation until it finishes for the first time.

The "Full Index" button populates a Queue which is periodically serviced by a worker. The worker will process the records and format for consumption by the Solr service. **This may take several hours to fully complete if the Hub has a lot of content.**

If an error with the worker occurs, a warning message such as this will appear.