Installing the Keystone server with the Federated plugin

Document Revision History

Version	Date	Description of change	Person
1	17-01-14	First public release	Kristy Siu
1.1	20 Feb 14	Removed known bug	David Chadwick
1.2	05-03-14	Added troubleshooting section and small addition to section 5.	Kristy Siu

Table of Contents

1. Introduction

This installation guide describes how to set up a Keystone server

a) To support federated access using an external Identity Provider (IdP)
b) To configure it with an external Keystone IdP
c) To configure it with a SAML IdP
d) To configure it with an ABFAB IdP

It uses the federated authentication plugin developed by the University of Kent to allow authentication by an external IdP. The software is tested using Ubuntu 12.04. It is assumed that the installation environment already has python 2.7, pip and git installed

2. Setting up Federated Keystone with DevStack

2.1 Modifying the stackrc and localrc files

The client developed by the University of Kent supports access to swift via Federated Keystone. When installing with Devstack (http://devstack.org/) the swift service is not enabled by default. To enable it, create or modify the file localrc to add the following line:

```
enable_service swift
```

To ensure the correct version of the Keystone server is installed when running the stack.sh script it is necessary to modify the stackrc file to point to the correct repository. The following is an example of the changes required to the stackrc file, whereby KEYSTONE_REPO is the address of the federated keystone repository and KEYSTONE_BRANCH is the branch name of the current version of the federated Keystone software:

```
# unified auth system (manages accounts/tokens)
KEYSTONE_REPO=https://github.com/kwss/keystone.git
KEYSTONE_BRANCH=fed-plugin-moonshot
```

2.1 Adding permissions to swift for mapped roles.

In order for swift to grant permission to users to create, modify or read the containers and files in the object store it is necessary to change the access policy of swift to include the roles assigned by the attribute mapping service. To do this you must modify the proxy-server.conf file. This file is usually located at /etc/swift/proxy-server.conf. Under the section [filter:keystoneauth] append the roles to the operator_roles line.

3. Install dependencies

3.1 Keystone

No addition dependencies to a standard Keystone installation are required to configure an external Keystone server as an Identity Provider to the Federated Keystone server.

3.2 SAML

- The packages libxml2-dev and libxmlsec1-dev should be installed using aptitude:

```
sudo apt-get install libxml2-dev libxmlsec1-dev
```

- An XML security library is required and can be obtained from:

https://pypi.python.org/packages/source/d/dm.xmlsec.binding/dm.xmlsec.binding-1.0b4.tar.gz

OR installed using pip

```
pip install dm.xmlsec.binding-1.0b4
```

3.3 ABFAB / Moonshot

In addition to the requirements for SAML the following dependencies are needed for an ABFAB / Moonshot installation.

- A modified python moonshot library is required and can be obtained from:

http://github.com/gierschv/pymoonshot

```
git clone http://github.com/gierschv/pymoonshot.git
cd pymoonshot/
sudo python setup.py install
```

- Add the moonshot repository to your repository sources:

```
echo "deb http://repository.project-moonshot.org/debian-moonshot sid main" > /etc/apt/sources.list.d/moonshot.list && wget -O - http://repository.project-moonshot.org/key.gpg | apt-key add - && apt-get update
```

You should use apt to install the following additional packages:

- Krb5-config
- Libkrb5-dev
- Linux-image-3.2.0-43-generic-pae
- libradsec0
- libradsec-dev
- Moonshot-gss-eap

```
apt-get install krb5-config libkrb5-dev libradsec0 libradsec-dev
```

- Install the moonshot dependencies:

```
apt-get install moonshot-gss-eap
```

- Add the mechanisms:

```
mkdir -p /usr/etc/gss
cat > /usr/etc/gss/mech <<EOF
eap-aes128 1.3.6.1.5.5.15.1.1.17 mech_eap.so
eap-aes256 1.3.6.1.5.5.15.1.1.18 mech_eap.so
EOF
```

In order to enable your moonshotted Keystone to contact your radius server you must add the following configuration file:

- `/etc/radsec.conf`

Example content:

```
realm gss-eap {
        type = UDP
        timeout = 5
        retries = 3
        server {
            hostname = "192.168.1.141"
            service = "1812"
            secret = "mysecret"
        }
    }
```

4. FreeRadius

Information about configuring FreeRadius can be found here:
http://freeradius.org/doc/

4.1 SAML Assertions Issued by FreeRadius

The FreeRadius server used to provide authentication should issue a SAML assertion containing a set of attributes which can be mapped (see Set up the Attribute Mapping and Attribute Issuing Policies). It is required that assertions contain a Subject element as well as the lifespan data contained in the Conditions attributes NotBefore and NotAfter.

5. Install Federated Keystone

This section only applies if you did not use Devstack to install the Openstack components. Use this section if you wish to install a Keystone server without other components or if your infrastructure requires each component to be installed separately. If you have used Devstack then you can skip to section 6.

The modified Keystone code can be found at:

https://github.com/kwss/keystone.git

Under the branch fed-plugin-moonshot

```
git clone https://github.com/kwss/keystone.git
cd keystone
git checkout fed-plugin-moonshot
git pull origin fed-plugin-moonshot
```

The standard installation instructions for Openstack Keystone should then be followed and can be found at:

http://docs.openstack.org/developer/keystone/installing.html

6. Set up config files

Keystone is configured using a configuration file called keystone.conf, it is normally located in /etc/keystone/. Full details of the configuration file can be found at

http://docs.openstack.org/trunk/openstack-compute/install/yum/content/keystone-conf-file.html

This section describes how to modify your configuration file to work with Federated Keystone.

In the directory ‘federated-docs’ that was downloaded in the previous section is an example configuration file for the keystone server. In the [auth] section there are extra options added to enable federated authentication. Firstly, the federated method should be enabled by appending this to the list of methods. Then the driver for the federated plugin should be set (see example below. Following this desired protocols should be enabled, SAML and Keystone are already enabled in the example file. To enable moonshot make sure abfab is included in the protocols option and that the correct module is specified by the abfab option. Here is an example of a config entry with all three protocols enabled:

```
[auth]
methods = password, token, federated
federated = keystone.auth.plugins.federated.Federated
protocols = saml, abfab, keystone
saml = keystone.auth.plugins.federated.protocol.saml.SAML
abfab = keystone.auth.plugins.federated.protocol.abfab.ABFAB
keystone = keystone.auth.plugins.federated.protocol.keystone.Keystone
```

7. Add Identity Provider

When adding identity providers to Federated Keystone the v2.0 API should be used because the keystone command line client has not been updated for v3 currently. To make administrating the server easier you can export the following environment variables where “mykeystoneserverip” is replaced with the address of your keystone server and “mytoken” is replaced with the admin token in your keystone.conf file:

```
OS_SERVICE_ENDPOINT=http://mykeystoneserverip:35357/v2.0
OS_SERVICE_TOKEN=mytoken
```

Requests made using cURL should include the admin token as a header:

```
-H {‘X-Auth-Token’: mytoken}
```

7.1 Keystone

7.1.1 UUID formatted tokens

In order to support UUID tokens you must provide an admin username and password combination that can be used to validate the tokens received from the remote Keystone server. You can provide this separately from, or alongside the certificates for handling PKI tokens (below).

7.1.2 PKI formatted tokens

The Keystone identity provider can be added to the keystone service catalog in the same way as a normal service, the endpoint however must be added using an HTTP post so that we can include the certificate details required to validate tokens issued by the Identity Provider as well as the Certificate Authority certificate of the Identity Provider. cURL (http://curl.haxx.se/) can be used for this (example below).

```
keystone service-create –name service-name –type idp.keystone –description “Keystone Identity
Provider”

curl –X POST –d ‘{“endpoint”:{“service_id”:”6e45a2”,
“publicurl”:”http://www.mykeystoneendpoint.com”,
“adminurl”:”http://www.mykeystoneendpoint.com”,
“internalurl”:”http://www.mykeystoneendpoint.com”, “certdata”:”base64encoded-certificate”,
“ca-cert”:”base64encoded-certificate”, “username”:”admin”, “password”: “mypass”}}’
http://mykeystoneendpoint.com:35357/v3/endpoints
```

7.2 SAML

The SAML identity provider can be added to the keystone service catalog in the same way as a normal service, the endpoint however must be added using an HTTP post so that we can include the certificate details required to validate assertions issued by the Identity Provider. cURL (http://curl.haxx.se/) can be used for this (example below).

```
keystone service-create –name service-name –type idp.saml –description “SAML Identity
Provider”

curl –X POST –d ‘{“endpoint”:{“service_id”:”6e45a2”,
“publicurl”:”http://www.mysamlendpoint.com”,  “adminurl”:”http://www.mysamlendpoint.com”,
“internalurl”:”http://www.mysamlendpoint.com”, “certdata”:”base64encoded-certificate”}}’
http://mykeystoneendpoint.com:35357/v3/endpoints
```

7.3 ABFAB / Moonshot

The moonshot identity provider can be added to the keystone service catalog in the same way as a normal service. Currently it is necessary to add an endpoint for the abfab service, but the given URLs are not used, so can be anything. (This requirement will be removed in a later version.)

```
keystone service-create --name service-name --type idp.abfab --description “ABFAB Identity
Provider”

keystone endpoint-create --service-id=7e427 publicurl=… --internalurl=… --adminurl=…
```

8. Set up the Attribute Mapping and Attribute Issuing Policies

The Attribute Issuing Policy says which IdPs are trusted to issue which identity attributes. Here is an example issuing policy, the rules in it specify that the Identity Provider with id 1 in the Keystone service catalog can issue the attribute role_type with the values student or staff and also the attribute uid with any value.

```
<IssuingPolicy>
        <Issuer service_id="1">
                <Attributes>
                        <Attribute type="role_type">
                                <AttributeValue>student</AttributeValue>
                                <AttributeValue>staff</AttributeValue>
                        </Attribute>
                        <Attribute type="uid">
                                <AttributeValue/>
                        </Attribute>
                </Attributes>
        </Issuer>
</IssuingPolicy>
```

The attribute mapping policy says which externally provided identity attributes should be mapped into which internal Keystone authorisation attributes. The ID attribute of the projects and roles should be used as the values for internal attributes.

Here is an example attribute mapping policy:

```
<AttributeMappings>

        <!-- ANY_OF -->
        <AttributeMapping>
                <ExternalAttributes>
                        <Attribute type="uid" requirement="any_value" />
                </ExternalAttributes>
                <InternalAttributes>
                        <Attribute type="project">
                                <AttributeValue>any_uid_project_ID</AttributeValue>
                        </Attribute>
                        <Attribute type="role">
                                <AttributeValue>any_uid_role_ID</AttributeValue>
                        </Attribute>
                </InternalAttributes>
        </AttributeMapping>

        <!-- ONE_OF -->
        <AttributeMapping>
                <ExternalAttributes>
                        <Attribute type="uid" requirement="one_of">
                                 <AttributeValue>david</AttributeValue>
                                <AttributeValue>bob</AttributeValue>
                        </Attribute>
                        <Attribute type="role_type" requirement="one_of">
                                 <AttributeValue>staff</AttributeValue>
                                <AttributeValue>student</AttributeValue>
                        </Attribute>
                </ExternalAttributes>
                <InternalAttributes>
                        <Attribute type="project">
                                <AttributeValue>
                                         student_or_staff_bob_or_david_project_ID
                                 </AttributeValue>
                        </Attribute>
                        <Attribute type="role">
                               <AttributeValue>
                                         student_or_staff_bob_or_david_role_ID
                                 </AttributeValue>
                        </Attribute>
                </InternalAttributes>
        </AttributeMapping>

        <!-- NOT_ANY_OF -->
        <AttributeMapping>
                <ExternalAttributes>
                        <Attribute type="uid" requirement="not_any_of">
                                <AttributeValue>david</AttributeValue>
                                <AttributeValue>bob</AttributeValue>
                        </Attribute>
                </ExternalAttributes>
                <InternalAttributes>
                        <Attribute type="project">
                                <AttributeValue>not_bob_or_david_project_ID</AttributeValue>
                        </Attribute>
                        <Attribute type="role">
                                <AttributeValue>not_bob_or_david_role_ID</AttributeValue>
                        </Attribute>
                </InternalAttributes>
        </AttributeMapping>
</AttributeMappings>
```

There are three types of rule:

- any_value

This type of rule on an attribute means that any value of this attribute type is valid for the rule to permit assignment of the mapped roles, projects and domains. Attribute values specified with this requirement will be ignored. The first mapping rule in the example says that anyone with a uid attribute (of any value) can be assigned the any_uid_role_ID on the any_uid__project_ID

- one_of

This type of rule on an attribute means that the user must possess one of the specified values of this attribute type for the rule to permit assignment of the mapped roles, projects and domains.
The second mapping rule says that anyone with the uid attribute with the attribute value ‘bob’ OR ‘david’ AND with the attribute role_type with the value ‘student’ OR ‘staff’ can be assigned the role ‘student_or_staff_bob_or_david_role_ID’ on the ‘student_or_staff_bob_or_david _project_ID’.

- not_any_of

This type of rule on an attribute means that a user must not possess this attribute for the rule to permit assignment of the mapped roles, projects and domains. If no value is specified then possession of the attribute with any value is sufficient for the mapping to not take effect, otherwise only users who match any of the specified value(s) will not have the mapping applied. The final rule says that anyone EXCEPT the user with the attribute uid=’bob’ or uid=’david’ can be assigned the ‘not_bob_or_david_role_ID’ on the ‘not_bob_or_david_project_ID’

9. Known Bugs

None at present

10. Troubleshooting

Problem	Solution
I receive a ‘HEADER TOO LONG’ error when trying to access Swift	Check in your swift configuration file for the max_header_size option and increase this until the error resolves. If you still receive the error after a large increase then you may be using an incompatible version of Swift. Download the latest Swift master branch instead.