Keystone VOMS module Documentation

Release 9.0.2.dev8

IFCA - CSIC

December 14, 2016

Contents

This module is intended to provide VOMS authentication to an OpenStack Keystone. It is designed to be integrated as an external authentication plugin, so that Keystone will preserve its original features and users will still be able to authenticate using any of the Keystone native mechanisms.

This documentation is based on a Keystone 9 (Mitaka) installation.

Attention: If you are upgrading from any priorversion, check the Upgrade nodes before proceeding.

- Only V2 authentication is supported in Mitaka so far.
- If you are using the 2014.1 (Icehouse) version, please check the Icehouse Documentation. Note that support in Icehouse is only for V2 authentication.
- If you are using the 2014.2 (Juno) version, please check the Juno Documentation. Note that support in Icehouse is only for V2 authentication.
- If you are using the 2015.1 (Kilo) version, please check the Kilo Documentation. Note that support in Kilo is only for V2 authentication.
- If you are using the 8.0.0 (Liberty) version, please check the Liberty Documentation. Note that support in Liberty is only for V2 authentication.

CHAPTER 1

User documentation

If you do not intend to install it, but rather authenticate against a VOMS service that is VOMS enabled, check the following link.

1.1 VOMS authentication client plugin

In order to facilitate the usage of the VOMS authentication with an OpenStack installation an authentication plugin for the nova client has been created. Please follow its instructions in order to use it.

1.2 Manual Usage

In order to get a token, you must post a JSON request in the body containing the following:

```
{
    "auth": {
        "voms": "true"
    }
}
```

In order to get a scoped token, use the following JSON document:

```
{
    "auth": {
        "voms": "true",
        "tenantNane": "TenantForTheVo",
    }
}
```

CHAPTER 2

Deploying a VOMS Authentication in Keystone

If you are a resource provider willing to deploy a VOMS-enabled keystone service, check the following documentation.

Attention: If you are upgrading from any priorversion, check the Upgrade nodes before proceeding.

Note: Default configuration files vary by distribution. You might need to add these sections and options rather than modifying existing sections and options. Also, an ellipsis (...) in the configuration snippets indicates potential default configuration options that you should retain.

2.1 VOMS Overview

The VOMS service can issue x509 proxies based on RFC 3820 (https://www.ietf.org/rfc/rfc3820.txt) by using the `-rfc` option in the comandline. Instead of using plain X.509 certificates this proxy can be used to authenticate against a properly configured Keystone server.

2.2 Keystone VOMS Overview

Follow this guide to enable your Keystone to be used with VOMS authentication. No modifications in the DB are needed, since it will be installed as an external plugin. Therefore, your Keystone will be usable with any other authentication mechanism that you had implemented (such as the native Keystone authentication).

This VOMS authentication module assumes that Keystone is working behind an http server as a WSGI application. SSL must be enabled in the http server.

Currently it only works with the V2 API of Keystone, a module compatible with the V3 API a work in progress.

2.3 How does it work?

SSL info is obtained from the request environment. The authentication module uses the VOMS library to check if the VOMS proxy is valid and if it is allowed in this server. The mapping between a VO, VO group and a keystone tenant is made in a configurable JSON file. For the moment there is no mapping for the Roles and/or Capabilities incoming from the VOMS credentials.

The mapped local tenant must exist in advance for a user to be authenticated. If the mapped tenant doest not exist, the authentication will fail. The same applies for the user, with the particularity that the backend is able to autocreate new incoming users if the `autocreate_users` is enabled in the configuration file and the authentication is sucessful (i.e. the proxy is accepted and it is valid). This option is disabled by default, but if you want to let all the users from a VO to get into your infrastructure you should consider enabling it. Once a user has been granted access, you can manage it as you will do with any other user in keystone (e.g. disable/enable, grant/revoke roles, etc.).

In order to get an unscoped token, you must POST to `/tokens`, with the following JSON document document in the request:

```
{
    "auth": {
        "voms": "true"
    }
}
```

This request should return you your an unscoped token. Next step is the discovery of your tenant (that may differ from the VO name). You have to use a GET request to `/tenants` passing the ID of your unscoped token (that you obtained before) in the `X-Auth-Token` header.

For further details, check the Test it! section.

2.4 Requirements

The Keystone VOMS authentication module requires some additional packages to be installed. Moreover, it requires that you run Keystone as a WSGI application behind an HTTP server (Apache will be used in this documentation, but any webserver could make it). Keystone project has deprecated eventlet, so you should be already running Keystone in such way.

- Keystone Mitaka.
- EUgridPMA CA certificates at the latest version.
- fetch-crl package.
- VOMS libraries.
- HTTP server with WSGI enabled.

2.4.1 EUgridPMA CA certificates and fetch-crl

You must have EUgridPMA certificates installed on its standard location (`/etc/grid-security/certificates`) and the `fetch-crl` package properly working so as have the CRLs up to date.

Ubuntu 14.04

Use these commands to install on Ubuntu:

```
$ wget -q -O - https://dist.eugridpma.info/distribution/igtf/current/GPG-KEY-EUGridPMA-RPM-3 | apt-ke
$ echo "deb http://repository.egi.eu/sw/production/cas/1/current egi-igtf core" \
  | tee --append /etc/apt/sources.list.d/egi-cas.list
$ apt-get update
$ apt-get install ca-policy-egi-core fetch-crl
$ fetch-crl
```

CentOS 7

Install CAs and fetch-crl with:

```
$ curl -L http://repository.egi.eu/sw/production/cas/1/current/repo-files/EGI-trustanchors.repo | suc
$ sudo yum install ca-policy-egi-core fetch-crl
```

2.4.2 VOMS libraries

You must install the VOMS libraries. Please install the `libvomsapi1` package in Debian/Ubuntu or `voms` package in RedHat/Fedora/ScientificLinux/etc:

```
$ apt-get install libvomsapi1
```

2.4.3 Apache Installation and Configuration

Note: Since Kilo, the keystone project deprecates Eventlet in favor of a WSGI server. This guide assumes that you already have a Keystone instance using the Apache HTTP server with mod_wsgi to serve Keystone requests on ports 5000 and 35357.

You need Keystone working under Apache WSGI with `mod_ssl` enabled. The standard installation starting on Liberty, guides through this setup, so during thise guide we will only set up the SSL part. To do so, enable the relevant module.

Ubuntu:

```
$ a2enmod ssl
```

CentOS:

```
$ yum install mod_ssl
```

Then add to your Apache Keystone WSGI configuration the SSL options as shown below. We assume that you have the CA certificates installed in the default location, otherwise you should adapt it to your needs, also check that the paths of the wsgi script are correct for your installation. The location of the configuration file for the keystone service under Apache depends on your distribution (e.g. in Ubuntu is `/etc/apache2/sites-available/wsgi-keystone.conf`, in CentOS 7 is `/etc/httpd/conf.d/wsgi-keystone.conf`). Note that you need a valid certificate for the http server (`SSLCertificateFile` and `SSLCertificateKeyFile`):

```
Listen 5000
WSGIDaemonProcess keystone user=keystone group=nogroup processes=8 threads=1
<VirtualHost _default_:5000>
    LogLevel     warn
    ErrorLog    ${APACHE_LOG_DIR}/error.log
    CustomLog   ${APACHE_LOG_DIR}/ssl_access.log combined

    SSLEngine               on
    SSLCertificateFile      /etc/ssl/certs/hostcert.pem
    SSLCertificateKeyFile   /etc/ssl/private/hostkey.pem
    SSLCACertificatePath    /etc/grid-security/certificates
    SSLCARevocationPath     /etc/grid-security/certificates
    SSLVerifyClient         optional
    SSLVerifyDepth          10
```

```
    SSLProtocol             all -SSLv2
    SSLCipherSuite          ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
    SSLOptions              +StdEnvVars +ExportCertData

    WSGIScriptAlias     /  /usr/bin/keystone-wsgi-public
    WSGIProcessGroup    keystone
</VirtualHost>

Listen 35357
WSGIDaemonProcess   keystoneapi user=keystone group=nogroup processes=8 threads=1
<VirtualHost _default_:35357>
    LogLevel    warn
    ErrorLog    ${APACHE_LOG_DIR}/error.log
    CustomLog   ${APACHE_LOG_DIR}/ssl_access.log combined

    SSLEngine               on
    SSLCertificateFile      /etc/ssl/certs/hostcert.pem
    SSLCertificateKeyFile   /etc/ssl/private/hostkey.pem
    SSLCACertificatePath    /etc/grid-security/certificates
    SSLCARevocationPath     /etc/grid-security/certificates
    SSLVerifyClient         optional
    SSLVerifyDepth          10
    SSLProtocol             all -SSLv2
    SSLCipherSuite          ALL:!ADH:!EXPORT:!SSLv2:RC4+RSA:+HIGH:+MEDIUM:+LOW
    SSLOptions              +StdEnvVars +ExportCertData

    WSGIScriptAlias     / /usr/bin/keystone-wsgi-admin
    WSGIProcessGroup    keystoneapi
</VirtualHost>
```

As you can see, the `SSLVerifyClient` is set to `optional`, so that people without a VOMS proxy can authenticate using their Keystone credentials.

Next, set the variable `OPENSSL_ALLOW_PROXY_CERTS` to `1` in your Apache environment configuration file (`/etc/apache2/envvars` in Debian/Ubuntu, `/etc/sysconfig/httpd` in CentOS) so that X.509 proxy certificates are accepted by OpenSSL. This is an important thing, so please double check that you have really enabled it.

Ubuntu:

```
$ echo "export OPENSSL_ALLOW_PROXY_CERTS=1" >> /etc/apache2/envvars
```

CentOS:

```
$ echo "OPENSSL_ALLOW_PROXY_CERTS=1" >> /etc/sysconfig/httpd
```

With the above configuration, and assuming that the Keystone host is `keystone.example.org` the endpoints will be as follow:

- `https://keystone.example.org:5000/` will be public and private endpoints, thus the Keystone URL will be `https://keystone.example.org:5000/v2.0`
- `https://keystone.example.org:35357/` will be administration endpoint, thus the Keystone URL will be `https://keystone.example.org:35357/v2.0`

2.4.4 Catalog

If you did not have Keystone running behind https your have to adjust your Keystone catalog so that the identity backend points to to the correct URLS as explained above. With the above configuration, these URLs will be:

- public URL: `https://keystone.example.org:5000/v2.0`
- admin URL: `https://keystone.example.org:35357/v2.0`
- internal URL: `https://keystone.example.org:5000/v2.0`

Note that the rest of the OpenStack configuration should be adjusted.

2.5 Upgrade notes

2.5.1 Upgrading to 2014.2 (Juno)

If you are upgrading from any prior version to the 2014.2 (juno) version, the VOMS filter in the API paste file (`/etc/keystone/keystone-paste.ini`) has changed from:

```
[filter:voms]
paste.filter_factory = keystone_voms:VomsAuthNMiddleware.factory
```

to:

```
[filter:voms]
paste.filter_factory = keystone_voms.core:VomsAuthNMiddleware.factory
```

2.6 VOMS module Installation

This module assumes that you are running the Keystone 9 (Mitaka) version.

2.6.1 Install the Keystone VOMS module

Install from Repositories

You can install Keystone VOMS from any of the repositories published in the AppDB. If you plan to install it like this, remove any prior version installed via pip and check that you are removing the old versions. If you did not install this module using pip, just ignore this step:

```
$ pip uninstall python-keystone-voms keystone-voms
```

Please, do not use the OpenSuse build service anymore and switch to the EGI AppDB repositories. Please visit the Keystone-VOMS product page where you can find the download page for all the available and supported versions.

Install from pip

With a running Keystone you can install the VOMS module with the following command (note the version range):

```
$ pip install 'keystone-voms>=9.0.0,<10.0.0'
```

Install from source

First, uninstall any old `python-keystone-voms` installation. This was the old name of the package and should be removed:

```
$ pip uninstall python-keystone-voms
```

With a running Keystone, simply install this egg. In the upper-level directory run `python setup.py install`:

```
$ git clone git://github.com/IFCA/keystone-voms.git -b stable/mitaka
$ cd keystone-voms
$ pip install .
```

2.6.2 Enable the Keystone VOMS module

The authentication module is a WSGI middleware that performs the authentication and passes the authenticated user down to keystone. Add the VOMS filter to your paste configuration file (`/etc/keystone/keystone-paste.ini` is the default one) First, add the VOMS filter as follows:

```
[filter:voms]
paste.filter_factory = keystone_voms.core:VomsAuthNMiddleware.factory
```

Then add this filter to the `public_api` pipeline for the version V2 of your API. Probably, you should add it before the `debug`, `ec2_extension`, `user_crud_extension` and `public_service` components:

```
[pipeline:public_api]
(...)
pipeline = sizelimit url_normalize build_auth_context token_auth admin_token_auth xml_body_v2 json_bo
```

Note that you may have a different pipeline. You don't need to replace your pipeline with the above, but just add the `voms` filter in the correct place.

2.7 VOMS module Configuration

2.7.1 VOMS configuration options

There are several new options in `/etc/keystone/keystone.conf` that are used to configure the VOMS identity behaviour. The default values should be OK for most installations, except the `autocreate_users` option. These optiones are under the `[voms]` section:

```
[voms]
vomsdir_path = /etc/grid-security/vomsdir
ca_path = /etc/grid-security/certificates
voms_policy = /etc/keystone/voms.json
vomsapi_lib = libvomsapi.so.1
autocreate_users = False
add_roles = False
user_roles = _member_
```

- `vomsdir_path`: Path storing the `.lsc` files.
- `ca_path`: Path where the CAs and CRLs are stored.
- `voms_policy`: JSON file containing the VO/tenant/role mapping.
- `vomsapi_lib`: Path to the voms library to use.
- `autocreate_users`: Whether a user should be autocreated if it does not exist.
- `add_roles`: Whether roles should be added to users or not.
- `user_roles`: list of role names to add to the users (if `add_roles` is `True`).

2.7.2 Allowed VOs

For each allowed VO, you need a subdirectory in `/etc/grid-security/vomsdir/` that contains the `.lsc` files of all truted VOMS servers for the given VO. The LSC files must be named as the fully qualified host name of the VOMS server with an `.lsc` extension, and they contain:

- First line: subject DN of the VOMS server host certificate.
- Second line: subject DN of the CA that issued the VOMS server host certificate.

So, for example, for the dteam VO the first should be:

```
$ cat /etc/grid-security/vomsdir/dteam/voms.hellasgrid.gr.lsc
/C=GR/O=HellasGrid/OU=hellasgrid.gr/CN=voms.hellasgrid.gr
/C=GR/O=HellasGrid/OU=Certification Authorities/CN=HellasGrid CA 2006
```

The dteam VO has two VOMS servers (see link above) so another file for the second server needs to be present as well:

```
$ cat /etc/grid-security/vomsdir/dteam/voms2.hellasgrid.gr.lsc
/C=GR/O=HellasGrid/OU=hellasgrid.gr/CN=voms2.hellasgrid.gr
/C=GR/O=HellasGrid/OU=Certification Authorities/CN=HellasGrid CA 2006
```

For more details, please check the following page How to configure VOMS LSC files. Note that you do not need to install the `.pem` certificate, just the `.lsc` file.

VO to local tenant mapping

The VO and VO group mapping to the local tenants is made in the JSON file `/etc/keystone/voms.json`. It is based on the VO name and VOMS proxy fqan:

```
{
    "voname": {
        "tenant": "local_tenant"
    }
}
```

For example for the dteam VO, it could be configured as:

```
{
    "dteam": {
        "tenant": "dteam"
    },
    "/dteam/NGI_IBERGRID": {
        "tenant": "dteam_ibergrid"
    }
}
```

If there are no matching FQANS but there is a VO name definition, the user will authenticate, therefore, a user making the following request:

```
{
    "auth": {
        "voms": "true",
        "tenantNane": "/dteam/NGI_IBERGRID",
    }
}
```

against the following configuration:

```
{
    "dteam": {
        "tenant": "dteam"
    }
}
```

will be sucessfully authenticated, because no FQAN matched, but the VO did.

2.8 Test it!

Once you have everything configured you can test it requesting a token using a valid VOMS proxy.

First get a valid VOMS proxy:

```
$ voms-proxy-init -voms <VOMS> -rfc
```

Then, get a unscoped token from the keystone server:

```
$ curl --cert $X509_USER_PROXY  -d '{"auth":{"voms": true}}' \
-H "Content-type: application/json" \
https://<keystone_host>:5000/v2.0/tokens
```

This will give you something like:

```
{
    "access": {
        "token": {
            "expires": "2011-08-10T17:45:22.838440",
            "id": "0eed0ced-4667-4221-a0b2-24c91f242b0b"
        }
    }
}
```

Use the token ID that you obtained, to get a list of the tenants that you are allowed to access:

```
$ curl -H "X-Auth-Token:0eed0ced-4667-4221-a0b2-24c91f242b0b" \
http://<keystone_host>:5000/v2.0/tenants
```

If this is sucessful, you should get something like:

```
 {
    "tenants_links": [],
    "tenants": [
        {
            "description": "Some Tenant",
            "enabled": true,
            "id": "999f045cb1ff4684a15ebb334af61461",
            "name": "TenantName"
        }
    ]
}
```

Identify the tenant, and request a scoped token:

```
$ curl --cert $X509_USER_PROXY  \
-d '{"auth":{"voms": true, "tenantName": "TenantName"}}' \
-H "Content-type: application/json" \
https://<keystone_host>:5000/v2.0/tokens
```

Finally, you should obtain your token:

```
{
    "access": {
            (...)
        },
        "serviceCatalog": [
                (...)
        ],
        "token": {
            "expires": "2013-07-30T12:16:23Z",
            "id": "ccb739df861e76a5a9039d21ec040a91",
            "issued_at": "2013-07-29T12:16:23.625426",
            "tenant": {
                "description": "Some Tenant",
                "enabled": true,
                "id": "999f045cb1ff4684a15ebb334af61461",
                "name": "TenantName"
            }
        },
        "user": {
            (...)
        }
    }
}
```

If everything is OK, you should be able to start using it.

2.9 Troubleshooting

2.9.1 Apache complains about issuer of certificate

You get something like:

```
Certificate Verification: Error (20): unable to get local issuer certificate
```

You probably missed to set the `OPENSSL_ALLOW_PROXY_CERTS` variable on the Apache environment

2.9.2 Error 14: Signature error

You have to double check that the `vomsdir_path` and `ca_path` configuration options (that default to `/etc/grid-security/vomsdir` and `/etc/grid-security/certificates` respectively) point to the correct path. Also ensure that the `.lsc` files have the right contents and that the CLRs are up to date.