Metadata aggregations how-to use pyff

Context

The goal of this how-to is to explain – technically – how to generate SAML Metadata Streams for your federation, in a very simple way. This how-to assumes that you already manage metadata by hand.
At the end of this how-to, you will be able to produce signed metadata for your federations.

This how-to is based on Pyff (https://pyff.readthedocs.io/, https://github.com/IdentityPython/pyFF), which uses YAML config files to generate metadata. Let's call those files "Pipelines".

Overview about pyFF

PyFF (python Federation Feeder) is a simple but complete SAML metadata aggregator.
Below are some of the key features:
• Fully customizable processing pipelines in yaml
• Easy to retrieve, analyze, transform, sign and publish SAML metadata
• Operate in batch or online mode (using embedded HTTP server) pyFF operates by setting up and running "pipelines". Pipelines are yaml files with a list of processing steps that are processed in order:

```
-  step 1
-  step 2
-  step 3
-  …
```

Each step represents a processing instruction. pyFF has a library of built-in instructions that include fetching local and remote metadata, xslt transforms, signing, validation and various forms of output and statistics. Processing steps are called pipes. A pipe can have arguments and options:

```
-  step [option]*:
   - argument1
   - argument2
   …
-  step [option]*:
   key1: value1
   key2: value2
   …
```

Below are the most common steps (i.e. pipes):

- The "load" statement retrieves (and parses) SAML metadata.
- The "select" statement is used to form an active document on which subsequent instructions operate.
- The "reginfo" statement sets registration info extension on EntityDescription element.
- The "finalize" statement sets validity and name of the metadata stream.
- The "sign" statement signs the output file.

- The “xslt” statement applies a XSL transformation.
- The “publish” statement tells where to put the result.
- The “stats” statement prints out some information about the current active document.

Install pyFF Prerequisites

CentOS 7 has already an installation of Python. But pyFF is developed using 3.6. So, we need to install Python 3 and make it as default.

1. Become root

```
sudo  su -
```

2. Check the version of Python installed

```
python -V
```

you should have: Python 2.7.5

3. Install the needed rpm repository for CentOS 7

```
yum install -y https://repo.ius.io/ius-release-el7.rpm
```

4. Install Python and pip. Pip is the package manager for Python packages and modules

```
yum install -y python36u python36u-libs python36u-devel python36u-pip
```

5. Upgrade pip to the latest version

```
pip3 install --upgrade pip
```

6. Use the version of Python that needs to be the default:

```
ln -fs /usr/bin/python3.6 /usr/bin/python
```

7. Check to confirm that Python3 (3.6.8) is now the default version.

```
python -V
```

Entities metadata collection

Although it's not mandatory, it's recommended to create a working directory (e.g. /usr/local/pyff) under which you maintain a tree structure for each federation you are managing. Below is a sample structure:

```
/base/path/for/your/metadata/files/
├── test-fede
│   ├── idps
│   │   ├── preprod-test-idp.xml
│   │   └── test-idp.xml
│   └── sps
│       ├── preprod-test-sp.xml
│       └── test-sp.xml
└── national-fede
    ├── idps
    │   ├── idp-university-X.xml
    │   └── idp-university-Y.xml
    └── sps
        ├── filesender.xml
        └── university-X-moodle.xml
```

In the following example you have 2 federations:

- A test federation
- A national federation

1. Install pyFF

```
pip3 install pyFF
```

2. Create your working directory

```
mkdir /usr/local/pyff
```

3. Create your first pipeline file

```
cd /usr/local/pyff
```

```
vim edugain-pipeline.yml
```

add =>

```
- load:
  - http://mds.edugain.org
- select:
- stats:
```

4. Run pyFF and provide the pipeline file as an argument (wait few seconds for the server to download eduGAIN metadata)

```
pyff edugain-pipeline.yml
```

If you get "command not found", you need to add /usr/local/bin to your user's path by running the command export PATH=$PATH:/usr/local/bin

You should get an output like the below with number IdPs AND SPs in eduGAIN:

```
---
total size: xxxx
selected: xxxx
idps: xxxx
sps: xxxx
---
```

This output confirms that you have successfully fetched, parsed, selected, and printed stats for the edugain metadata feed. In the next sections, you will learn how to generate signed SAML metadata streams for your federation(s).

Generate SAML metadata stream using pyFF

Let us consider now that you are managing your national federation "mynren" and you need to generate its SAML metadata stream using pyFF. You are planning to register first the Idp and the SP (configured in the previous sessions) and test the setup.

1. Start with creating your sub-directories structure for mynren federation

```
mkdir /usr/local/pyff/mynren
```

```
mkdir /usr/local/pyff/mynren/idps
```

```
mkdir /usr/local/pyff/mynren/sps
```

2. The metadata you generate MUST be signed. You need a self-signed public/private key pair. Here's how to generate one, with 10 years validity:

Enter your VM hostname vm-0000**XX**.vm.geant.org when asked about the "Common Name" _

```
openssl req -nodes -x509 -newkey rsa:4096 -keyout sign.key -out sign.crt -
days 3650
```

check that sign.crt and sign.key are generated:

```
ll /usr/local/pyff
```

3. Collect the metadata files of the IdPs or SPs you want to add into your federation

```
cp /opt/shibboleth-idp/metadata/idp-metadata.xml
/usr/local/pyff/mynren/idps/
```

You SP metadata data will be loaded through https. As an alternative, you can copy the xml into the sps directory.

4. Create mynren pipeline yml file

vim /usr/local/pyff/mynren-pipeline.yml

```
- load:
   - https://vm-0000XX.vm.geant.org/Shibboleth.sso/Metadata
   - /usr/local/pyff/mynren/idps/idp-metadata.xml
- select:
- reginfo:
     authority: https://vm-0000XX.vm.geant.org/
     policy:
        en: https://vm-0000XX.vm.geant.org/policy.html
- xslt:
     stylesheet: pp.xsl
- finalize:
    cacheDuration: PT5H
    validUntil: P10D
- sign:
    key: /usr/local/pyff/sign.key
    cert: /usr/local/pyff/sign.crt
- publish: /var/www/html/vm-0000XX.vm.geant.org/mynren.xml
- stats:
```

Explanations:

Section	Description
load	Enumerates all metadata files to include in the federation
select	Filtering
reginfo	Sets registration info extension on EntityDescription element
finalize	Sets validity and name of the metadata stream
sign	Signs the file
xslt	Applies a XSL transformation. pp.xsl is builtin Pyff and makes pretty print
publish	Tells where to put the result

5. Generate the aggregate and signed metadata of your federation

cd /usr/local/pyff

pyff mynren-pipeline.yml

You should receive an output like the below:

```
---
total size: 2
  selected: 2
      idps: 1
       sps: 1
---
    -
```

6. Check that your metadata are generated and published (it should contain IdPs and SPs you configured):

Open this URL in your browser https://vm-**0000XX**.vm.geant.org/mynren.xml

Conclusion and Next steps (not mandatory for this training)

Following this how-to, you have a way of generating signed, syntactically correct metadata files.

You can configure your IdP and SP to trust your new federation, and login from your IdP to your SP.

The next steps to take should be:

- Setup a cron job to periodically generate your metadata files
- Collect new metadata files:
    - Copy them in the correct folder
    - Update the pipeline accordingly
    - Keep track of
        - When did I got this metadata file?
        - Who gave it to me?
        - In which federation should they be?

Good luck!