The FreeRadius Moonshot SAML module

This package provides two different FreeRadius modules, intended to be used in Moonshot environments. They extend the functionality of a Moonshot IDP and/or RPP by adding advanced SAML assertion handling capabilities, including dynamic generation of SAML assertions (on the IDP), manipulation of SAML attributes (on both the IDP and RPP), and authorisation based on SAML attributes (on the RPP).

Contents:

- 1. Installation
- 2. Configuration and use
  - 2.1. IDP module
    - 2.1.1. Enabling the module
    - 2.1.2. Configuring the module
      - 2.1.2.1. Attribute sources
      - 2.1.2.2. Attribute filters
    - 2.1.3. A complete example of IDP configuration
  - 2.2. RPP module
    - 2.2.1. Enabling the module
    - 2.2.2. Configuring the module
      - 2.2.2.1. Authorization rules
    - 2.2.3. A complete example of RPP configuration

1. Installation

You can install this package by using our repositories:

Debian / Ubuntu

```
apt-get install freeradius-moonshot-saml
```

CentOS / RHEL / SL 6 and 7

```
apt-get install freeradius-moonshot-saml
```

CentOS / RHEL / SL 8

```
apt-get install freeradius-moonshot-saml
yum -y install python2-pip python2-devel openldap-devel gcc
pip2 install python-ldap
```

Alpine

```
apk add freeradius-moonshot-saml
apk add py2-pip python2-dev openldap-dev gcc musl-dev
pip2 install python-ldap
```

2. Configuration and use

Once the package is installed, the individual modules must be explicitly enabled and configured to use their functionalities.

2.1. IDP module

2.1.1. Enabling the module

To do so, just create a soft link of the module into `$FR_CONFIG_PATH/mods-enabled`, where `$FR_CONFIG_PATH` is the path where FreeRadius configuration files are found (eg. `/etc/raddb or /etc/freeradius`).

```
ln -s $FR_CONFIG_PATH/mods-available/abfab_idp $FR_CONFIG_PATH/mods-enabled
```

Once the module has been enabled, FreeRadius needs to know where to use it. This is done by adding its name to the `post-auth` section of the `$FR_CONFIG_PATH/sites-enabled/inner_tunnel` file.

Since the module needs to have access to the `GSS-Acceptor-*` RADIUS attributes from the `Access-Request` message in order to know who the RP is, it is required to set the `copy_request_to_tunnel=yes` option in the `ttls` and `peap` sections of the `$FR_CONFIG_PATH/mods-enabled/eap` file.

Similarly, the module generates `SAML-AAA-Assertion` RADIUS attributes that need to be included in the final `Access-Accept` RADIUS message. The easiest way of enabling this is by looking for the following stanza in the `$FR_CONFIG_PATH/sites-enabled/inner_tunnel` file and making sure that `if (1)` is set.

```
    #  Instead of "use_tunneled_reply", change this "if (0)" to an
    #  "if (1)".
    #
    if (1) {
```

2.1.2. Configuring the module

Once the module is installed and enabled, you can tune up its behaviour in the `/etc/abfab_idp.conf` file. This file is required and contains a single JSON object with the following elements:

- `assertion_issuer`. Value to be used for the `<Issuer/>` SAML element in the generated SAML assertion. Default: `"https://abfab_idp"`.
- `assertion_lifetime`. The module will use this value to calculate the expiration time of the SAML assertion. The lifetime is expressed in minutes. Default: `60`.
- `attribute_sources`. A list of attribute sources (see below) from where end user attributes will be retrieved. Default: `[]`.
- `attribute_filters`. A list of attribute filters (see below) to be applied to the retrieved attributes before building the SAML Assertion. Default: `[]`.

2.1.2.1. Attribute sources

Attribute sources are defined as JSON objects, where the `type` element defines its type (one of `ldap`, `sqlite`, or `static`).

The `ldap` attribute source retrieves user attributes from a LDAP server. It defines the following additional configuration elements:

- `ldap_uri`. URI of the LDAP server. Example: `"ldap://ldap.exaple.org:389"`.
- `ldap_login`. LDAP login string. Optional. Example: `"cn=admin,dc=whatever,dc=whatever"`.
- `ldap_password`. LDAP password string. Optional. Example: `"supersecretkey"`.
- `ldap_basedn`. LDAP base DN for the queries. Example: `"dc=users,dc=whatever,dc=whatever"`.
- `ldap_mapping`. Mapping to determine how RADIUS `User-Name` attribute (denoted as `%u`) is used to retrieve the LDAP information. Example: `"(mail=%u)"`.
- `ldap_timeout`. Maximum time to wait for a response, expressed in seconds. Default: 2.

The `sqlite` attribute source retrieves user attributes from a SQLite3 database. It defines the following configuration elements:

- `sqlite_db`. Location of the SQLite3 attribute database. Example: `attributes.sqlite`. This database must have a table called `attributes` with the following schema:

```
CREATE TABLE attributes ('username' TEXT NOT NULL,
                         'name' TEXT NOT NULL,
                         'name_format' TEXT NOT NULL,
                         'value' TEXT NOT NULL,
                         PRIMARY KEY('username', 'name'));
```

The `static` attribute source always produce the same attribute for all the queries. Its main purpose is to enforce a specific attribute that makes sense at an organization level (e.g. organization name). It defines the following configuration elements:

- `static_name`: Attribute name. Example: `"organization-name"`.
- `static_name_format`: Attribute name format. Default: `"urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified"`.
- `static_value`. Attribute value. Example: `"Hogwarts"`.

2.1.2.2. Attribute filters

Attribute filters allow performing basic transformations to the list of attributes. They are defined as JSON objects, where the `type` element defines its type (one of `blacklist`, `template`, or `combine`). They also share the following configuration elements:

- `rp_name`. The filter will only be applied if the textual name of the RP (as in `service@hostname`) matches this regular expression. Default: `".*"`.
- `idp_name`. The filter will only be applied if the realm name of the IDP matches this regular expression. Default: `".*"`.

The `blacklist` attribute filter allows removing undesired attributes. It defines the following configuration elements:

- `exclude`. A list of regular expressions. Attributes whose name match any of these will be excluded from the list. Default: `[]`.
- `include`. A list of regular expressions. Attributes whose name match any of these will not be excluded even if they match the `exclude` list. Default: `[]`.

The `template` attribute filter allows creating a new attribute based on parts of the value of another attribute. It defines the following configuration elements:

- `source`. Name of the source attribute. Example: `"full-name"`.
- `destination`. Name of the new attribute to be created. Default: `"new_template_attribute"`.
- `re`. Regular expression to match parts of the attribute value. Example: `"^(.*) (.*)$"`
- `template`. Template (as in Python string templates) to build the new attribute. Example: `"Surname: {1}. Name: {0}"`.

The `combine` attribute filter allows creating a new attribute based on the values of several attributes. It defines the following configuration elements:

- `sources`. List of source attribute names. Example: `["surname", "name"]`.
- `destination`. Name of the new attribute to be created. Default: `"new_combined_attribute"`.
- `template`. Template (as in Python string templates) to build the new attribute. Example: `"Name: {1}. Surname: {0}"`.

2.1.3. A complete example of IDP configuration

```
{
    "assertion_issuer": "https://testidp.org/",
    "attribute_sources": [
        {
            "type": "ldap",
            "ldap_uri": "ldap://ldap.um.es:389",
            "ldap_basedn": "dc=usuarios,dc=um,dc=es",
            "ldap_mapping": "(mail=%u)"
        },
        {
            "type": "static",
            "static_name": "organization",
            "static_value": "University of Murcia"
        }
    ],
    "attribute_filters": [
        {
            "rp_name": ".*@server.org",
            "type": "blacklist",
            "exclude": [".*"],
            "include": ["organization"]
        }
    ]
}
```

2.2. RPP module

2.2.1. Enabling the module

To do so, just create a soft link of the module into `$FR_CONFIG_PATH/mods-enabled`.

```
ln -s $FR_CONFIG_PATH/mods-available/abfab_rpp $FR_CONFIG_PATH/mods-enabled
```

Once the module has been enabled, FreeRadius needs to know where to use it. This is done by adding its name to the `post-auth` section of the `$FR_CONFIG_PATH/sites-enabled/abfab-tr-idp` file.

2.2.2. Configuring the module

Once the module is installed and enabled, you can tune up its behaviour in the `/etc/abfab_rpp.conf` file. This file is required and contains a single JSON object with the following elements:

- `attribute_filters`. A list of attribute filters (as described for the IDP module) to be applied to the SAML Assertion before proxying it. Default: `[]`.
- `authorization_rules`. A list of authorization rules (see below) that the SAML Assertion must comply in order to grant access to the RP. Default: `[]`.

2.2.2.1. Authorization rules

Authorization rules allow defining a set of attributes and their corresponding values that MUST be present on the SAML Assertion in order to grant the user access to the RP. In case all rules are not satisfied, the `Access-Accept` RADIUS message is replaced with an `Access-Reject` one.

Authorization rules are defined as JSON objects, with the following configuration elements:

- `name`. Regular expression that must match attribute name. Default: `".*"`.
- `nameformat`. Regular expression that must match attribute name format. Default: `".*"`.
- `value`. Regular expression that must match attribute value. Default: `".*"`.

2.2.3. A complete example of RPP configuration

```
{
    "attribute_filters": [
        {
            "type": "combine",
            "sources": ["name", "surname"],
            "template": "{0} {1}",
            "destination": "fullname"
        },
        {
            "idp_name": "idp.*",
            "type": "template",
            "source": "fullname",
            "re": "^(\\S)* (.*)$",
            "template": "{1}",
            "destination": "surname_v2"
        },
        {
            "type": "blacklist",
            "exclude": [
                "undesired_attr_1",
                "undesired_prefix_1.*"
            ],
            "include": ["undesired_prefix_1_2"]
        }
    ],
    "authorization_rules": [
        {
            "name": "surname",
            "value": "PEREZ.*"
        },
        {
            "name": "entitlement",
            "value": "professor"
        }
    ]
}
```