Opensips Radius 配置

This page has been visited 8195 times.

Table of Content (hide)

RADIUS is a client/server networking protocol that provides centralized Authentication, Authorization, and Accounting management for clients to connect and use a network service.

After completing this tutorial, you will be able to:

- make a basic configuration for the RADIUS server
- make a basic configuration for the RADIUS client
- use accounting and authentication features provided by **OpenSIPS**
- add and use custom attributes in RADIUS messages
- send custom authentication requests and fetch data from responses
- send custom accounting requests and inspect the server logs

---

1. Installing RADIUS Server

The first step is to install FreeRADIUS server. On a Debian based system this can be done using:

```
# apt-get install freeradius
```

or you can go to FreeRADIUS homepage , download the binaries and follow the instructions there. For more information, check out FreeRADIUS Installation .

---

2. Installing RADIUS Client

On a Debian based system you can install RADIUS client using:

```
 # apt-get install libradiusclient-ng2
```

or you can go to radiusclient-ng homepage , download the binaries and follow the instructions there.

---

3. Tips and tricks to configure RADIUS Client and Server for Authentication

There are a few things you have to make sure you do properly, so that you can use RADIUS authentication. The accounting features need no special configuration. For the server configuration, you may need to be a super user.

3.1 Make sure both client and server know and share the same SIP dictionary

The master dictionary path is specified for the RADIUS client in the */etc/radiusclient-ng/radiusclient.conf* file. The radiusclient comes with particular dictionaries located in the */etc/radiusclient-ng/* folder, including the **SIP** specific dictionary called *dictionary.sip* , but by default they are not included. You can add it manually at the end of the master dictionary, or you can include it to the master dictionary file by inserting the following line:

```
$INCLUDE <desired_dictionary_path>
```

where the <desired_dictionary_path> should be the path for the RADIUS dictionary, which is by default */etc/radiusclient-ng/dictionary.sip* .

The master dictionary file for the freeRADIUS server is */etc/freeradius/dictionary* . It references the pre-defined dictionary files included with the server that are located, by default, in the */usr/share/freeradius/* folder. Also, the master dictionary does not include the **SIP** specific dictionary entries. You must include the same **SIP** dictionary the client does with the same insert method used by the client ($INCLUDE).

3.2 Configure RADIUS server and RADIUS client to share the same secret

For the radiusclient, you need to specify in the *radiusclient-ng.conf* file the path for the file containing the secrets. By default, this path is */etc/radiusclient-ng/servers*.

For the FreeRADIUS server, the file you need to configure is */etc/freeradius/clients.conf* .

3.3 Set up FreeRADIUS to handle digest authentication requests

In order to do that, you just need to uncomment the digest lines in both **authenticate{}** and **authorize{}** sections of the site in the */etc/freeradius/sites-available/default* file. This is the default site, you may want to change that.

3.4 Configure the /etc/freeradius/users file to match the user you want to authenticate and the attributes you want to be returned

For example:

```
testdig Cleartext-Password:= opensips.cfg
    Reply-Message = "OpenSIPS Rules!",
    SIP-AVP = "sems:ann-account_locked",
    Sip-Rpid = sip:+407433360111@opensips.org

DEFAULT Auth-Type := Accept
    Reply-Message = "OpenSIPS Rules!",
    SIP-AVP = "sems:ann-account_locked",
    Sip-Rpid = sip:+407433360111@opensips.org
```

This is an example of a freeRADIUS users configuration file that contains two rules: one for a user called **testdig** and one for a **default** user. When a RADIUS request is received, the users are tested sequentially until a match can be made. DEFAULT matches any user. If a DEFAULT user is not specified and no match is made, the request is ignored.

---

4. How to use RADIUS support from the OpenSIPS configuration script

There are two types of RADIUS messages to send: accounting messages and authentication and authorization messages. **OpenSIPS** provides support for both types of RADIUS messages.

4.1 Accounting

Accounting refers to the tracking of the calls made by users. This information may be used for management, billing, or other purposes. Therefore, when the server receives an accounting message, it doesn't send any AVPs in the Radius reply, and so no attribute can be fetched from a reply, since the message contains no AVPs.

The accounting logs held by the server can be inspected here: */var/log/freeradius/radacct/* . They are sorted by date and have suggestive names.

OpenSIPS accounting module overview

The **OpenSIPS** module that supports accounting is the ACC module. Its responsibility is to account transactions information to different back-ends, including a generic AAA back-end that has a RADIUS implementation.

OpenSIPS accounting module configuration

To enable the RADIUS accounting support, you have to set the script parameter called **aaa_url** for the ACC module like this:

```
modparam("acc", "aaa_url", "radius:/etc/radiusclient-ng/radiusclient.conf")
```

The **aaa_url** contains the name of the AAA protocol used, in this case RADIUS and the location of the configuration file of this protocol, separated by “:”.

There are other optional flags and parameters you may want to use, depending on your purpose. For example, you may want to do accounting for a specific **SIP** message, or for a whole transaction. For more information about that and more, check ACC AAA flags .

A special feature of the accounting module is the optional **aaa_extra** parameter. Its functionality is to account extra values via AAA (in our case, RADIUS). This functionality is a limited version of the custom RADIUS queries, which will be presented later on. You can set this parameter like this:

```
modparam("acc", "aaa_extra", "via=$hdr(Via[*]); email=$avp(s:email);
Bcontact=$ct / reply")
```

OpenSIPS accounting module usage

This is an example of usage of the AAA function provided by the ACC module in which a comment is appended.

```
...
acc_aaa_request("Some comment");
...
```

This function sends a predefined list of AVPs. To see what these AVPs are, after you configure the server, the client and the script properly, use this function and then inspect the server logs.

4.2 Authentication and authorization

Authentication refers to verifying the client's identity. To do that, the FreeRADIUS server checks the */etc/freeradius/users* to see if the user is known. If so, the FreeRADIUS checks the credentials received and sends a reply containing the result of the authentication. When performing authentication, the AAA server may include in the response additional information, i.e. those attributes you have previously configured for that user. Only these attributes can be inspected from the reply message. If the use doesn't match any entry in the users file, the request is ignored.

OpenSIPS RADIUS authentication module overview

The **OpenSIPS** module that supports authentication and authorization is the AUTH_AAA module. Its responsibility is to authenticate users using a AAA back-end that has a RADIUS implementation. The proxy will simply pass along the credentials to the freeRADIUS server and expect the result of authentication.

OpenSIPS RADIUS authentication module configuration

Similar to the accounting module, to enable authentication, you have to set the **aaa_url** from the script.

```
modparam("auth_aaa", "aaa_url", "radius:/etc/radiusclient-ng/radiusclient.conf")
```

There are other optional flags and parameters you may want to use, depending on your purpose. For more information about them, check AUTH_AAA flags.

OpenSIPS authentication module usage

Here are some examples of usage for the functions exported by the AUTH_AAA module:

```
...
if (!aaa_www_authorize("opensips.org")) {
    www_challenge("opensips.org", "1");
};
...
...
if (!aaa_proxy_authorize("")) {   # Realm and URI user will be auto-generated
    proxy_challenge("", "1");
};
...
if (!aaa_proxy_authorize("$pd", "$pU")) { # Realm and URI user are taken
    proxy_challenge("$pd", "1");         # from P-Preferred-Identity
};                                           # header field
...
```

4.3 Custom accounting and authentication

OpenSIPS has introduced a new feature that allows any type of RADIUS queries to be yielded directly from the script, and also, to inspect RADIUS replies for certain attributes. The module that handles these operations is the AAA_RADIUS module.

To enable this module, you simply have to set the **radius_config** parameter, containing the path for the radiusclient library configuration file. For example:

```
modparam("aaa_radius", "radius_config", "/etc/radiusclient-
ng/radiusclient.conf")
```

Custom RADIUS AVPs

Since all other RADIUS operations use predefined attributes, the custom queries provided by the AAA_MODULE have the ability to use custom attributes also.

In addition to the standard attributes, RADIUS supports also custom attributes, that can be vendor proprietary, or manually configured. According to RFC2865, the value of an attribute is represented on 1 octet. Hence, the maximum value for an attribute will be 255. Using the Vendor Specific Attribute encapsulation, the vendor proprietary attributes are allowed to have values greater than 255.

For vendor specific attribute encapsulation, check out: RADIUS dictionary manual . The following part of this tutorial will handle only non-vendor specific attributes.

Due to the fact that for an attribute to be correctly interpreted by both server and client, for setting up a custom attribute, you have to edit the dictionary files for both of them.

For the RADIUS server, you have to add to one of the dictionaries included, preferably to the **SIP** specific dictionary a new entry with the desired attribute name, value and type. The value must be smaller than 256. The same thing has to be done for the radius client dictionary file also. Make sure the entries are identical for the server and for the client.

How to specify a query

Both accounting and authenticating custom queries use a common way of specification, that is a **set**. They are used when building custom RADIUS requests (set of input RADIUS AVPs) or when fetching data from the RADIUS reply (set of output RADIUS AVPs).

The format for a set definition is the following:

```
 set_name = ( attribute_name1 = var1 [, attribute_name2 = var2 ]* )
```

The left-hand side of the assignment must be an attribute name known by the RADIUS dictionary. The right-hand side of the assignment must be a script pseudo variable or a script AVP. For more information about them see CookBooks - Scripting Variables. For example:

```
modparam("aaa_radius","sets","set1 = (User-Name=$var(usr), Sip-Group =
$var(grp), Service-Type = $var(type)) ")
```

When a query is made, the SIP AVPs will be expanded and the the pairs (attribute_name, var) will be added to the RADIUS message.

Custom accounting queries

To send an accounting custom query, the function to be used is **radius_send_acct** . The function takes only one parameter that represents the name of the set that contains the list of attributes and pvars that will form the accounting request. Only one set is needed as a parameter because no AVPs can be extracted from the accounting replies. The set must be defined using the "sets" exported parameter. For example:

```
...
radius_send_acct(“set1”);
...
```

Custom authentication queries

To send an accounting custom query, the function to be used is **radius_send_auth** . The function takes two parameters: the name of the set that contains the list of attributes and pvars that will form the authentication request and the name of the set that contains the list of attributes and pvars that will be extracted form the authentication reply. The sets must be defined using the "sets" exported parameter. For example:

```
...
radius_send_auth(“set1”,”set2”);
...
```

---

5. Useful links:

http://wiki.freeradius.org/

http://www.opensips.org/html/docs/modules/devel/aaa_radius.html

http://www.opensips.org/html/docs/modules/devel/acc.html

http://www.opensips.org/html/docs/modules/devel/auth_aaa.html

http://www.ietf.org/rfc/rfc2865.txt

http://www.ietf.org/rfc/rfc2866.txt

http://voiprookie.blogspot.com/2009/04/freeradius-and-mysql.html