ConfD integration with FreeRADIUS for Authentication

ConfD's External Authentication

The most common and easiest way to work with ConfD is to use ConfD's built-in authentication and authorization mechanisms in which all the AAA-related information is being stored in CDB. The next most common way is to use PAM, which allows the same set of users and groups as those of the Linux host itself to have access to ConfD. However, there are times when it is necessary due to system requirements to store all the authentication data remotely, such as on a remote RADIUS server. This remote authentication server typically not only stores the users and their passwords, but also the group information.

This application note describes how to integrate ConfD with FreeRADIUS running on the Ubuntu environment. FreeRADIUS is an open source project supplying a feature-rich implementation of the RADIUS protocol with its various enhancements (http://www.freeradius.org). In addition to the configuration of ConfD, both the installation and configuration of FreeRADIUS followed by a sample authentication shell script will also be described.

Enabling External Authentication in confd.conf

In order to use external authentication with ConfD, the following XML blocks in confd.conf will need to be configured.

```
<confdConfig>
  <aaa>
    <authOrder>externalAuthentication localAuthentication</authOrder>
    <externalAuthentication>
        <enabled>true</enabled>
        <executable>./auth.sh</executable>
    </externalAuthentication>
</aaa>
</confdConfig>
```

/confdConfig/aaa/authOrder is used to define the order in which ConfD will follow to authenticate a user. The default order is “localAuthentication pam externalAuthentication”. Since we would like to use external authentication here, we will change the order to “externalAuthentication localAuthentication”. If the user credentials don’t exist in external authentication, then local authentication will be consulted. As we are not working with PAM at this time, we will take it out of the authOrder setting.

Setting /confdConfig/aaa/externalAuthentication/enabled to true will enable external authentication.

/confdConfig/aaa/externalAuthentication/executable is used to define the name of the executable to be invoked to authenticate a user. The executable will receive the username and the clear text password on its standard input in the format “[${USER};${PASS};]\n”. For example, if user is “bob” and password is “secret”, the executable will receive the string “[bob;secret;]” followed by a newline on its standard input. The program must parse this line from ConfD.

Refer to the confd.conf(5) man page for more information on other AAA-related settings.

External Authentication Executable

The task of the external authentication program as defined in confd.conf is to authenticate the user and also provide the user to groups mapping information to ConfD. For example, the program could be a RADIUS client which utilizes some proprietary vendor attributes to retrieve the groups of the user from the RADIUS server. If authentication is successful, the program should write “accept“ followed by a space-separated list of groups the user is a member of, and additional information as described below.

For example, if user “bob” attempts to login over SSH using the password “secret”, and external authentication is enabled, ConfD will invoke the configured executable and write “[bob;secret;]\n” on the stdin stream for the executable. Assuming that Bob is a member of the “admin” and the “lamers” groups, the program should write “accept admin lamers $uid $gid $supplementary_gids $HOME\n” on its standard output and then exit.

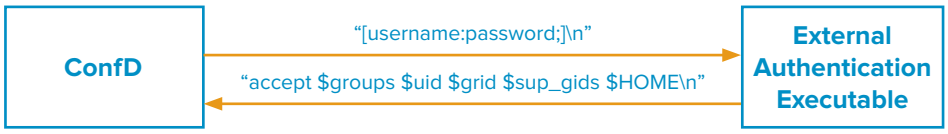
Thus the format of the output from an external authentication program when authentication is successful should be:

“accept $groups $uid $gid $supplementary_gids $HOME\n”

Where

- $groups is a space separated list of the group names the user is a member of.
- $uid is the UNIX integer user id ConfD should use as default when executing commands for this user.
- $gid is the UNIX integer group id ConfD should use as default when executing commands for this user.

- $supplementary_gids is a (possibly empty) space separated list of additional UNIX group ids the user is also a member of.
- $HOME is the directory which should be used as HOME for this user when ConfD executes commands on behalf of this user.



If authentication failed, the program should write "reject" or "abort", possibly followed by a reason for the rejection, and a trailing newline. For example, "reject Bad password\n" or just "abort\n". The difference between "reject" and "abort" is that with "reject", ConfD will try subsequent mechanisms configured for /confdConfig/aaa/authOrder in confd.conf (if any), while with "abort", the authentication fails immediately. Thus "abort" can prevent subsequent mechanisms from being tried, but when external authentication is the last mechanism (as in the default order), it has the same effect as "reject". Refer to the "External authentication" section of "The AAA infrastructure" chapter in the ConfD User Guide for more discussion on this.

Install FreeRADIUS from pre-built binary package

The pre-built FreeRADIUS package can be installed by using the following command on Ubuntu:

```
$ sudo apt-get install freeradius
```

The version of FreeRADIUS being used here is 2.2.8 built on April 5, 2016 running on Ubuntu 16.04 Desktop version.

Configuring FreeRADIUS with proprietary vendor attributes

In order for FreeRADIUS to be able to respond with the group information as required by ConfD, its dictionary file will need to be configured similar to the following:

1. Ensure that you are root or use sudo to edit the configuration files
2. Add the following line to /usr/share/freeradius/dictionary:

   $INCLUDE dictionary.tailf
3. Create a file called /usr/share/freeradius/dictionary.tailf with the following contents:

```
# -*- text -*-
##########################################################
#
#     Tail-f ConfD
#
#     $Id$
#
##########################################################

VENDOR               Tailf                           88

BEGIN-VENDOR         Tailf

ATTRIBUTE            Tailf-Group-Name       10      string
ATTRIBUTE            Tailf-Group-ID         11      integer
ATTRIBUTE            Tailf-User-ID          12      integer
ATTRIBUTE            Tailf-Home-Dir         13      string

END-VENDOR Tailf
```

Make sure that the VENDOR number (88 here) isn't being used by any other dictionary files in your FreeRADIUS setup by checking all the other vendor specific dictionary files. Otherwise, pick a different unique number.

The vendor specific attributes that are defined here correspond to the arguments that are required to be passed back to ConfD upon a successful authentication request by the external authentication program. Refer to FreeRADIUS man page on dictionary for further information.

Configuring FreeRADIUS users information

The following setup assumes that you will be using the default aaa_init. xml that comes with ConfD's example set. The aaa_init.xml file is being described in more detail in the "Populating AAA using CDB" section of "The AAA infrastructure" chapter in the ConfD User Guide. This aaa_init.xml file is by default copied to the CDB directory by the ConfD install scripts. It defines two users, admin and oper with passwords set to admin and oper respectively.

Since localAuthentication is being configured as the lower priority authentication mechanism in the authOrder setting confd.conf in our earlier section, these local users in aaa_init.xml will be checked for a possible match by ConfD upon a reject response from the external authentication executable. There is also the corresponding admin and oper groups defined along with their authorization rules. These same groups and authorization rules are being used in the setup here. Refer to the "Group Membership" section in "The AAA infrastructure" chapter of the ConfD User Guide for more information on ConfD's group based AAA authorization model.

FreeRADIUS is typically set up by modifying its configuration files that are located under /etc/freeradius.

1. Ensure that you are root or use sudo to edit the configuration files
2. FreeRADIUS includes a default client called localhost. This client can be used by client programs on the localhost to help with troubleshooting and testing. Confirm that the following entry exists in the clients.conf file:

   client localhost {
   ipaddr = 127.0.0.1
   secret = testing123
   require_message_authenticator = no
   nastype = other
   }

3. Define admin as a FreeRADIUS test user. Add the following lines at the top of the users file. Make sure the second and third lines are indented by a single tab character:

   “admin” Cleartext-Password := “admin”
   Tailf-Group-Name = “admin”,
   Tailf-Group-ID = 2,
   Tailf-User-ID = 3,
   Tailf-Home-Dir = “/usr/admin”,
   Reply-Message = “Hello, %{User-Name}”

You are free to choose any other username and password to be assigned to the admin group instead of admin/admin.

Performing a simple test using radtest

To make all of the previous configuration changes take effect, FreeRADIUS needs to be restarted. In order to make debugging easier, it is recommended to stop and restart freeradius with the –X option.

Check to see if freeradius has indeed been stopped by the first command using the ps command (ps aux | grep freeradius) before executing the second command. If it hasn’t been stopped, you will need to force it to shutdown by

```
$ sudo /etc/init.d/freeradius stop
$ sudo freeradius -X
```

using “kill -9 $(pidof freeradius)”.

1. Ensure that FreeRADIUS has started correctly by confirming that the last line on your screen says the following:

   Ready to process requests.

2. Authenticate the admin user using the following command:

```
$ radtest admin admin 127.0.0.1 100 testing123
```

Radtest is a RADIUS client test tool that comes with FreeRADIUS.
The first argument is the user name (admin).
The second argument is the password (admin).
The third argument is the server address (127.0.0.1).
The fourth argument is the nas-port-number (100). This value isn't important here but is required to be present.
The fifth argument is the shared secret (testing123) for the client as defined in clients.conf.

3. The debug output of the FreeRADIUS server will show how the Access-Request packet arrives and how the FreeRADIUS server responds with the Access-Accept packet.

4. Radtest will also show the response from the FreeRADIUS server as follows:

Sending Access-Request of id 151 to 127.0.0.1 port 1812
User-Name = "admin"
User-Password = "admin"
NAS-IP-Address = 127.0.1.1
NAS-Port = 100
Message-Authenticator = 0x00000000000000000000000000000000

rad_recv: Access-Accept packet from host 127.0.0.1 port 1812, id=151, length=89
Tailf-Group-Name = "admin"
Tailf-Group-ID = 2
Tailf-User-ID = 3
Tailf-Home-Dir = "/usr/admin"
Reply-Message = "Hello, admin"

External Authentication Shell script using radtest

Here is a sample shell script that first extracts the user name and password from the stdin and pass them into a radtest request. It then parses the response from radtest in order to be able to write the proper stdout response back to ConfD.

```
#!/bin/sh
#
# Reads username and password information from ConfD and invokes RADIUS
to
# perform external authentication
#
# input format: [username;password]
#

read line
username=`echo $line | cut -d '[' -f2 | cut -d ';' -f1`
password=`echo $line | cut -d '[' -f2 | cut -d ';' -f2`
radtest $username $password 127.0.0.1 100 testing123 | sed -n '7,$p' |
{
      read line
      RESPONSE="Access-Accept"
      if echo "$line" | grep -q "$RESPONSE"; then
              #processing line with group_name info
              read line
              group_name=`echo $line | cut -d'=' -f2 | sed -e
's/^[[:space:]]*//'`
```

```
            #processing line with group_id info
            read line
            group_id=`echo $line | cut -d'=' -f2 | sed -e
's/^[[:space:]]*//'`

            #processing line with user_id info
            read line
            user_id=`echo $line | cut -d'=' -f2 | sed -e
's/^[[:space:]]*//'`

            #processing line with user_id info
            read line
            home_dir=`echo $line | cut -d'=' -f2 | sed -e
's/^[[:space:]]*//'`

            echo "accept $group_name $group_id $user_id $home_dir
else
            echo "reject bad username and/or password"
      fi
}
```

The above sample script isn't meant to be production quality. On the contrary, a lot of assumptions are being made in order to simplify it. There is only 1 group being assigned to each authenticated user. All of the vendor specific attributes from the RADIUS responses always appear in the same order and their names are not checked.

In addition to shell scripts, the external authentication executable can also be written as a C program.

Running ConfD with External Authentication using FreeRADIUS

To test the script from the previous section, you can modify any one of the examples in the ConfD's example set with the confd.conf settings according to the "enabling External Authentication in confd.conf" section of this document. Make a copy of the script in the previous section, save it as auth.sh and put it under one of the example directories. Make sure to change the permission of auth.sh to be an executable ("chmod a+x auth.sh").

A simple test to run after the example has been started with "make start" is as follows:

```
$ netconf-console --hello
```

The default username and password as used by netconf-console is admin/admin. You can monitor the FreeRADIUS server's debug output for the RADIUS protocol exchanges. Any errors in the responses from the external authentication program will be logged in confd.log. Any successful logins will be logged in audit.log. You can also test for the rejected case by supplying the incorrect username and/or password using the --user and --password options of netconf-console.

tail-f

www.tail-f.com info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40

US Headquarters

5201 Great American
Pkwy Suite 320
Santa Clara, CA 95054
+1 408 466 4241

Japanese Distributor

A.I. Corporation
Iijima Bldg., 2-25-2
Nishi-Gotanda
Shinagawa-ku
Tokyo, 141-0031
Japan
+81 3 3493 7981

©2013 Tail-f systems. All rights reserved.