freeradius-idp

Setting up FreeRADIUS

This section describes how to set up FreeRADIUS for an IdP. It assumes that you have already executed the configuration steps for the eduroam SP configuration of FreeRADIUS. We will expand that configuration to turn FreeRADIUS into a simple IdP. N.B.: even if you are going to have an IdP-only installation, the eduroam SP configuration for FreeRADIUS is still the exact same. You just don't define any own Access Point clients in clients.conf.

Adding IdP support in FreeRADIUS needs several steps to be executed:

- a TLS server certificate needs to be created for EAP methods to work
- the desired EAP types need to be configured.
- the virtual server eduroam needs to be instructed to do tunneled EAP authentication
- a user database needs to be linked to the FreeRADIUS instance to authenticate the users
- a realm needs to be marked as to-be-authenticated-locally in the configuration
- the server needs to be prepared to process incoming requests *from* the upstream FLR server

These steps are explained in detail below. For the user database, this example will use a "flat file" with usernames and passwords. The Goodies section contains examples for MySQL and other types of backend databases.

TLS server certificate

While it is possible to buy and install a commercial TLS certificate, this is neither necessary (the trust settings of web-browser stores don't apply for EAP, so there are no "recognised" CAs) nor prudent (a commercial CA issues many certificates, and uncautious users might be tempted to accept other certificates from that same CA).

We suggest to create an own certificate. FreeRADIUS makes this very easy by providing an automatic script for that purpose. Execute the

```
/etc/raddb/certs/bootstrap
```

script. It will generate certificates which are suited for EAP authentication, and name them so that the server can find them immediately without further configuration. Later, for the supplicant configuration, you will need to include the generated CA certificate into your supplicant configurations.

EAP type configuration

The file /etc/raddb/eap.conf defines how EAP authentication is to be executed. The shipped configuration file is not adequate for eduroam use; it enabled EAP-MD5 and LEAP, for example; which are not suitable as eduroam EAP types. Use the following content for eap.conf instead. It enables PEAP and TTLS:

```
eap {
                default_eap_type = peap
                timer_expire     = 60
                ignore_unknown_eap_types = no
                cisco_accounting_username_bug = no

                tls {
                        certdir = ${confdir}/certs
                        cadir = ${confdir}/certs
                        private_key_password = whatever
                        private_key_file = ${certdir}/server.key
                        certificate_file = ${certdir}/server.pem
                        CA_file = ${cadir}/ca.pem
                        dh_file = ${certdir}/dh
                        random_file = /dev/urandom
                        fragment_size = 1024
                        include_length = yes
                        check_crl = no
                        cipher_list = "DEFAULT"
                }

                ttls {
                        default_eap_type = mschapv2
                        copy_request_to_tunnel = yes
                        use_tunneled_reply = yes
                        virtual_server = "eduroam-inner-tunnel"
                }

                peap {
                        default_eap_type = mschapv2
                        copy_request_to_tunnel = yes
                        use_tunneled_reply = yes
                        virtual_server = "eduroam-inner-tunnel"
                }

                mschapv2 {
                }

        }
```

A common question regarding this definition is: "why is TLS also configured? I don't want it, can I disable it?" The answer is: the TTLS and PEAP sections depend on the tls stanza for the definition of which server certificates to use. You cannot delete the stanza, but that doesn't mean you can't effectively disable TLS: the tls stanza contains the ca_file parameter. Only clients with a TLS client certificate from this CA will be accepted. We have just created a brand-new CA with the "bootstrap" script. Simply don't issue nor distribute any client certificates from this CA, then nobody will be able to log in with EAP-TLS. Note that in newer versions of FreeRADIUS (>3.0.14) there is a new tls-config section that allows you to configure the common TLS configuration without configuring the TLS EAP type. The config above is backwards compatable, but if you want to take advantage of the new section you can replace the name of the "tls" block above with "tls-config tls-common" and then reference it from each EAP type with "tls = tls-common" (the example eap config shows you how to do this).

Another question is regarding the mschapv2 section. For all practical purposes, the easy answer is that it is a piece of magic and needs to be there for PEAP to work. If you are curious regarding the gory details, please let us know.

Note that one parameter for both the ttls and peap stanza is "virtual_server = eduroam-inner-tunnel". This means that the inner EAP authentication will be carried out in this other virtual server, which we will define later.

Virtual server eduroam: enable EAP, make Operator-Name conditional

Compared to the eduroam SP config, you need to additionally mention the "eap" module in both the authorize and authenticate stanza of the file /etc/raddb/sites-enabled/eduroam so that your server can process EAP requests from your own userbase.

You should also make sure to only tag those incoming requests with the Operator-Name attribute which actually originate from your own WiFi gear - as an IdP, your own users roaming elsewhere will also be processed, but they should not carry your own Operator-Name. For the purposes of this wiki, let's assume that you are connected to one FLR server, and it is defined in your clients.conf with the shortname "antarctica-flr-1" (see below for the exact definition).

It will then look like the following:

```
authorize {
                if ("%{client:shortname}" != "antarctica-flr-1") {
                   update request {
                           Operator-Name := "1yourdomain.tld"
                            # the literal number "1" above is an important prefix! Do not change it!
                   }
                }
                auth_log
                suffix
                eap
        }

authenticate {
       eap
}
```

Virtual server eduroam-inner-tunnel

When the eap module has started with an authentication, it will first establish a TLS tunnel; this is done by enabling the module in the previous "eduroam" virtual server. After the TLS tunnel is established, the content (i.e. the tunneled authentication) is processed separately in this new virtual server. Create the file in /etc/raddb/sites-enabled/eduroam-inner-tunnel and give it the following content:

```
server eduroam-inner-tunnel {

authorize {
        auth_log
        eap
        files
        mschap
        pap
}

authenticate {
        Auth-Type PAP {
                pap
        }
        Auth-Type MS-CHAP {
                mschap
        }
        eap
}

post-auth {
        reply_log
        Post-Auth-Type REJECT {
                reply_log
        }

}
}
```

Let's revisit the modules which this virtual server executes one after another:

- auth_log: logs the incoming packet to the file system. This is needed to fulfill the eduroam SP logging requirements. Note that this log *may* contain the user's cleartext password if TTLS-PAP is used. You can log the packet with omitted User-Password attribute if you prefer; see the "Goodies" section for more details).
- eap: if the EAP authentication contains another EAP instance inside, the module will decode it. This is the case for PEAP.
- files: this module tries to find out the authoritative password for the user by looking up the username in the file
- mschap: this module is in effect only if PEAP-MSCHAPv2 or TTLS-MSCHAPv2 is used. It will mark the packet as to be authenticated with MS-CHAP algorithms later.
- pap: this module is in effect only if TTLS-PAP is used. It will mark the packet as to be authenticated with PAP alogrithms later.
- reply_log: logs the reply packet to the file system

User database: flat file

By default, the "files" module will use information in the file

```
/etc/raddb/users
```

for authenticating users. This file has a straightforward format

```
icecold@group1.aq       Cleartext-Password := "snowwhite"

otheruser@group1.aq     Cleartext-Password := "swordfish"
```

Local authentication for your realm

In the SP configuration, all requests were unconditionally forwarded to upstream. We will need to revisit the file "proxy.conf" and mark one realm to NOT proxy. In this example, we will use "@group1.aq" as the local authentication realm. Simply add the following stanza immediately preceeding the "DEFAULT" realm:

```
realm group1.aq {
        nostrip
}
```

Since the stanza doesn't contain a server pool to proxy to, this realm won't be proxied and instead authenticated locally. This stanza works only for users who correctly use the full username format "user123@group1.aq" for their eduroam login.

If the IdP and SP are colocated, it is possible to *locally* also accept users who erronuously omitted their realm (just "user123"). This is NOT permitted by the eduroam policy (read 6.3.2 bullet 6 under AAA Servers of the current service definition document: "The outer EAP identities (and with it, RADIUS User-Name attributes) for the IdP MUST be in the format of arbitrary@realm"). Allowing this also requires further configuration and it is strongly discouraged, because it will give such users a "halfways-working" experience: they will be able to use eduroam when on their own IdP's campus, because no routing information needs to be evaluated, but their account will fail at all other locations. Therefore, this guide does not include instructions for that kind of setup.

Processing incoming requests

As an eduroam IdP, your users can go to other eduroam hotspots around the globe. They will still be authenticated at your server. In these roaming cases, your upstream FLR servers will send Access-Requests to your server. Surprisingly, it is very simple to configure that: these upstream servers are simply clients - just like an Access Point. So, simply add client stanzas for your FLR servers into clients.conf:

```
 client antarctica-flr-1 {
        ipaddr                          = 172.20.1.2
        netmask                         = 32
        secret                          = secretstuff
        require_message_authenticator   = yes
        shortname                       = antarctica-flr-1
        nastype                         = other
        virtual_server                  = eduroam
}
```

CUI for eduroam IdP

To use the Chargeable-User-Identity (CUI) you must already use the Operator-Name attribute. This documentation is only for FreeRADIUS 3.0.X release.

Modify the log module

Edit "eduroam_cui_log" file in the mods-available/ subdirectory and add the following lines to your virtual inner server :

```
...
linelog cui_inner_log {
#    filename = syslog
    filename = ${logdir}/radius.log
    format = ""
    reference = "inner_auth_log.%{%{reply:Packet-Type}:-format}"
    inner_auth_log {
        Access-Accept = "%t : eduroam-inner-auth#VISINST=%{request:Operator-Name}#USER=%{User-Name}#CSI=%{%
{Calling-Station-Id}:-Unknown Caller Id}#NAS=%{%{Called-Station-Id}:-Unknown Access Point}#CUI=%{%{%{reply:
Chargeable-User-Identity}:-%{outer.reply:Chargeable-User-Identity}}:-Local User}#RESULT=OK#"
        Access-Reject = "%t : eduroam-inner-auth#VISINST=%{request:Operator-Name}#USER=%{User-Name}#CSI=%{%
{Calling-Station-Id}:-Unknown Caller Id}#NAS=%{%{Called-Station-Id}:-Unknown Access Point}#CUI=%{%{%{reply:
Chargeable-User-Identity}:-%{outer.reply:Chargeable-User-Identity}}:-Local User}#RESULT=FAIL#"
    }
}
```

Use policy and module in your eduroam-inner-tunnel virtual server

Add 'cui-inner' (policy already defined, you don't need to change it) and 'cui_inner_log' in post-auth section :

```
server eduroam-inner-tunnel {
...
               post-auth {
                               reply_log
                               cui_inner_log
                               cui-inner
                               Post-Auth-Type REJECT {
                                       reply_log
                                       cui_inner_log
                               }
               }
...
}
```

That's it! Now your server is prepared for eduroam IdP operation! You can add users to your "database" by amending the "users" file; if you do, you will unfortunately have to restart FreeRADIUS so that it picks up the change.

Goodies

Omitting User-Password in inner authentication logs

By default, the "detail" modules log every attribute as it was received. For inner authentications with TTLS-PAP, this means that the attribute "User-Password" with the user's perceived password will be logged. This is often considered harmful. You can deactivate it by blacklisting the attribute in the auth_log module in /etc/raddb/modules/auth_log:

```
detail auth_detail {
  ...
  suppress {
    User-Password
  }}
```

adding VLAN assignment attributes

You can mark every user with a VLAN where he should be put into. This is done by assigning "reply items" to the user in the authentication database. In our flat file example, reply attributes are in a separate line, indented by a tab. To put our two example users into VLANs 17 and 42, respectively, the entries would look like the following:

```
icecold@group1.aq       Cleartext-Password := "snowwhite"
                        Tunnel-Type             := VLAN,
                        Tunnel-Medium-Type      := IEEE-802,
                        Tunnel-Private-Group-ID := 17

otheruser@group1.aq     Cleartext-Password := "swordfish"
                        Tunnel-Type             := VLAN,
                        Tunnel-Medium-Type      := IEEE-802,
                        Tunnel-Private-Group-ID := 42
```

Using SQL as user database backend

Using a flat file as in our example scales very poorly. You can use arbitrary database backends instead; the FreeRADIUS documentation can give you an overview. If you wish to use SQL, changing our example configuration is very easy: simply replace the "files" line in eduroam-inner-tunnel:authorize with "sql". You'll need to specify the connection details for your SQL backend in the corresponding module (/etc/raddb/modules/sql).

The schema which FreeRADIUS uses to store user information is similarly structured to the "users" file: a table radcheck holds the check items (i.e. the username and password), and the radreply table contains the reply items (for example VLAN memberships, as explained above).

Mandating or forbidding use of anonymous outer identity

eduroam at large supports anonymous outer identities for user logins. It is at the discretion of eduroam IdPs whether they want to

- mandate that their users use an anonymous outer identity
- forbid their users to use an anonymous outer identity
- be agnostic in that respect

Configuring any one of the three choices is done with only a few lines of configuration. The easiest choice is being agnostic: no configuration is necessary, since there is no link between the inner and outer User-Name attribute in FreeRADIUS.

If you want to mandate the use of anonymous outer identities, the recommended way is using the identity "@realm" (i.e. the part left of the @ sign should be empty). You can enforce that only this outer User-Name is allowed to proceed to EAP authentication by adding the following to the authenticate section:

```
if ( User-Name != "@realm" ) {
      fail
}
```

If you want to forbid usage of anonymous outer identities, you can do this by comparing the two presented User-Name attributes of the outer and inner authentication. You can only do this in the eduroam-inner-tunnel virtual server obviously, since only that server has access to the inner identity. Put the following into the "authenticate" section of eduroam-inner-tunnel:

```
if ( User-Name != outer.User-Name ) {
     fail
}
```

More information

Eduroam-in-a-box web configuration tool:http://eduroam.sourceforge.net