freeradius-sp

FreeRADIUS is a very versatile and freely available RADIUS server under the GPL license. Setting up FreeRADIUS as an SP is a rather straightforward task, since it merely needs to forward requests from NASes to other RADIUS servers. In particular, it does not need to authenticate users. The following configuration enables your FreeRADIUS server to be an eduroam SP. At the same time, it is the baseline from which to establish an eduroam IdP configuration, if that is envisaged for a later stage.

1.1.1.1. Version information

This document is in migration from FreeRADIUS 2 to FreeRADIUS 3. We recommend using the last available version of the stable FreeRADIUS 3 branch. It's easy to compile version 3 (and create packages) if your distribution doesn't provide recent packages. (On Ubuntu/Debian with "make deb" for instance and "rpmbuild -ba redhat/freeradius.spec" should help you on Red Hat based systems.)

Some of the filesystem paths changed between version 2 and 3. The /etc/raddb/modules directory is now split between /etc/raddb/mods-available and /etc /raddb/mods-enabled, plus some of the configuration can be found in /etc/raddb/mods-config. Note that when a module isn't called from the rest of the configuration, placing it in mods-enabled doesn't mean it's active: only that it's available in the rest of your configuration.

1.1.1.2. Installation

FreeRADIUS is written in C and can be compiled with the usual UNIX compilation sequence. After unpacking the source into a directory of your choice, do

```
./configure --prefix=<your preferred install dir> --sysconfdir=<your preferred configuration base dir>
make
make install
```

In the examples below, we assume the installation is done for --prefix=/usr/local/freeradius/ and the configuration dir is --sysconfdir=/etc

1.1.1.3. Sample config directory

1.1.1.4. Base configuration / logging / F-Ticks

The main configuration file is /etc/raddb/radiusd.conf; it does not require many changes from the shipped default.

The following lines are important for eduroam operation: a server status probing mechanism called Status-Server is enabled in the security section. Make sure the config file contains the following security stanza

```
security {
           max_attributes = 200
           reject_delay = 0
           status_server = yes
}

proxy_requests      = yes
```

(From the default distribution, only reject_delay needs to be changed.)

FreeRADIUS is capable of both IPv4 and IPv6. By default, both are enabled in the listen {} section of sites-enabled/default so we'll duplicate them in our new sites-enabled/eduroam configuration. (The listen {} directives used to be in /etc/raddb/radiusd.conf for FreeRADIUS 2.) You can leave out the IPv6 part if your server shouldn't do IPv6.

The logic in the server is defined by activating modules in a certain order. These modules are separately defined in the /etc/raddb/mods-enabled/ subdirectory (and configured in /etc/raddb/mods-config/ where applicable). The order of activation of these modules is defined in so-called virtual servers, which are defined in the /etc/raddb/sites-enabled/ directory. For our eduroam SP purposes, we only need one virtual server "eduroam" and call very few of the modules. It needs to contain as a minimum:

```
server eduroam {

        listen {
                type = "auth"
                ipaddr = *
                port = 0
        }
        listen {
                type = "acct"
                ipaddr = *
                port = 0
        }
        listen {
                type = "auth"
                ipv6addr = ::
                port = 0
        }
        listen {
                type = "acct"
                ipv6addr = ::
                port = 0
        }

        authorize {
                # only use filter_username from version > 3.0.7 on
                filter_username
                                update request {
                        Operator-Name := "1yourdomain.tld"
                                                # the literal number "1" above is an important prefix! Do not
change it!
                }
                # if you want detailed logging
                auth_log
                suffix
        }

        authenticate {
        }

        preacct {
                suffix
        }

        accounting {
        }

        post-auth {
                # if you want detailed logging
                reply_log
                Post-Auth-Type REJECT {
                        reply_log
                }
        }

        pre-proxy {
                # if you want detailed logging
                pre_proxy_log
                if("%{Packet-Type}" != "Accounting-Request") {
                        attr_filter.pre-proxy
                }
        }

        post-proxy {
                # if you want detailed logging
                post_proxy_log
                attr_filter.post-proxy
        }
}
```

The multitude of sections in this above configuration is often confusing to new-comers. The order of execution when proxying a request are:

```
authorize  authenticate  pre-proxy
```

Then, the packet is proxied to an upstream server. When the reply comes back, the execution continues:

```
post-proxy  post-auth
```

Every stanza contains names of modules to be executed. Let's revisit them one after another:

- auth_log: logs the incoming packet to the file system. This is needed to fulfill the eduroam SP logging requirements.
- suffix: inspects the packet to look for an eduroam style realm (separated by the @ sign)
- pre_proxy_log: logs the packet to the file system again. Attributes that were added during the inspection process before are then visible to the administrator - great for debugging
- attr_filter.pre-proxy: strips unwanted attributes off of the request before sending the request to upstream
- post_proxy_log: logs the reply packet to the file system - as received by upstream
- attr_filter.post-proxy: strips unwanted attributes off of the reply, prior to sending it back to the Access Points (VLAN attributes in particular!)
- reply_log: logs the reply packet after attribute filtering to the file system

The paths where the logs are written to, and the files with the list of permitted attributes for filtering, are defined in the corresponding module definitions in /etc/raddb/modules/<name-of-module>.

If attr_filter.pre-proxy is enabled (as per the example above), then by default Operator-Name and Calling-Station-Id are stripped from the proxied request. In order for them not to be removed, add the attributes to /etc/raddb/attrs.pre-proxy (FreeRADIUS 2) or /etc/raddb/mods-config/attr_filter/pre-proxy (FreeRADIUS 3). This is a more sensible default for eduroam:

```
DEFAULT
        User-Name =* ANY,
        EAP-Message =* ANY,
        Message-Authenticator =* ANY,
        NAS-IP-Address =* ANY,
        NAS-Identifier =* ANY,
        State =* ANY,
        Proxy-State =* ANY,
        Calling-Station-Id =* ANY,
        Called-Station-Id =* ANY,
        Operator-Name =* ANY
```

Since the eduroam SP with this configuration will statically use RADIUS to its upstream federation-level server, activation of F-Ticks reporting is not strictly necessary. It is thus described only in the "Goodies" section below.

1.1.1.5. Client definition

FreeRADIUS defines the connected RADIUS clients in the file /etc/raddb/clients.conf. This file needs to hold all your connected Access Points (and/or wired eduroam-enabled switches, if you have these instead of Access Points). You set a shared secret for each client and define these in the config file as follows:

```
 client antarctica-access-point-1 {
    ipaddr                        = 172.25.1.55
    netmask                       = 32
    secret                        = yoursecret12345
    shortname                     = southpole-11g
    virtual_server                = eduroam
    require_message_authenticator = yes
 }
```

There are more (optional) settings for clients; please consult the comments in clients.conf for more detail. One option, the "virtual_server" one, enables your RADIUS server to serve more purposes than only eduroam: you can define several other virtual servers for other RADIUS purposes, and link clients to these. That is beyond the scope of this documentation, though.

If you want to connect your clients over IPv6, the syntax is only slightly different:

```
 client antarctica-access-point-2 {
    ipv6addr                      = 2001:db8:1:789::56
    netmask                       = 128
    secret                        = yoursecretABCDE
    shortname                     = southpole-11n
    virtual_server                = eduroam
    require_message_authenticator = yes
}
```

1.1.1.6. Request forwarding

FreeRADIUS contains a wealth of options to define how requests are forwarded. These options are defined in the file /etc/raddb/proxy.conf. For a single eduroam SP, these may seem overkill, but the required definitions for that purpose are rather static. Assuming you have two upstream servers to forward requests to, the following configuration will set these up - you only need to change the IP addresses and shared secrets in home_server stanzas. The realm NULL will reject authentication requests missing an @ sign, for example Windows always first tries its domain\hostname to authenticate when connecting the first time to eduroam. This authentication would otherwise be sent upstream to the realm "~.+$", which causes delays and is unneeded.

```
proxy server {
        default_fallback        = no
}

home_server antarctica-flr-1 {
        type                    = auth+acct
        ipaddr                  = 172.20.1.2
        port                    = 1812
        secret                  = secretstuff
        status_check            = status-server
}

home_server antarctica-flr-2 {
        type                    = auth+acct
        ipaddr                  = 172.25.9.3
        port                    = 1812
        secret                  = secretstuff
        status_check            = status-server
}

home_server_pool EDUROAM {
        type                    = fail-over
        home_server             = antarctica-flr-1
        home_server             = antarctica-flr-2
}

realm NULL {
    virtual_server  = auth-reject
    nostrip
}

realm "~.+$" {
        pool                    = EDUROAM
        nostrip
}
```

1.1.1.7. Goodies

1.1.1.7.1. Running FreeRADIUS as non-root user

The RADIUS protocol runs on ports >1023, which means it can be started entirely in unprivileged mode on UNIX-like systems. You can easily achieve that by

- creating a user "radiusd" and group "radiusd"
- giving all configuration files in /etc/raddb ownerships for that user radiusd + group radiusd
- changing these two parameters in /etc/raddb/radiusd.conf:

```
user  = radiusd
group = radiusd
```

1.1.1.7.2. F-Ticks

F-Ticks is using syslog to deliver user login statistics. You can enable syslog logging for login events by defining a *linelog* module. In the /etc/raddb /modules/ subdirectory, create a new file "f_ticks":

```
linelog f_ticks {
       filename = syslog
       #syslog_facility = local0
       #syslog_severity = info
       format = ""
       reference = "f_ticks.%{%{reply:Packet-Type}:-format}"
       f_ticks {
              Access-Accept = "F-TICKS/eduroam/1.0#REALM=%{Realm}#VISCOUNTRY=YOUR-TLD#VISINST=%{Operator-Name}
#CSI=%{Calling-Station-Id}#RESULT=OK#"
              Access-Reject = "F-TICKS/eduroam/1.0#REALM=%{Realm}#VISCOUNTRY=YOUR-TLD#VISINST=%{Operator-Name}
#CSI=%{Calling-Station-Id}#RESULT=FAIL#"
       }
}
```

Note that you have to adapt VISCOUNTRY to the country you are in (eg. set YOUR-TLD to "LU"), and VISINST to an identifier for your hotspot - which in this example is already set to the Operator-Name attribute. You can set the syslog facility and severity to help forward these ticks to the right place.

You need to enable this new module in the post-auth section of your virtual server eduroam:

```
post-auth {
                # if you want detailed logging
                reply_log
                f_ticks
                Post-Auth-Type REJECT {
                        # if you want detailed logging
                        reply_log
                        f_ticks
                }
        }
```

This way, appropriate loglines will be logged into your local syslog instance. If you want to forward your ticks to the statistics system, please get in touch with your NRO to get to know the syslog destination and configure your syslog daemon to forward the log line correspondingly.

Please note that the file proxy.conf may need your attention: FreeRADIUS' handling of the "DEFAULT" realm changed slightly between 2.1.9 and 2.1.10: previously, it would fill %{Realm} with the actual realm (e.g. "education.lu"), but after the change, it would use the literal "DEFAULT". It is not helpful to generate ticks with REALM=DEFAULT.

If you were using DEFAULT before, and now notice that ticks are sent incorrectly, the mitigation is to use a regular expression instead of DEFAULT - because for realm statements with regular expressions, also the most recent versions still substitute with the actual realm.

You would need to delete the DEFAULT realm and replace it with the following regular expression realm statement *at the end of your proxy.conf*:

```
realm "~.+$" {
...
}
```

1.1.1.8. CUI for eduroam SP

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.

1.1.1.8.1. Create a log module

By default the CUI is not logged, you have to use the FreeRADIUS *linelog* module to get a log. In the mods-available/ subdirectory, create a new file "eduroam_cui_log" :

```
linelog cui_log {
#    filename = syslog
    filename = ${logdir}/radius.log
    format = ""
    reference = "auth_log.%{%{reply:Packet-Type}:-format}"
    auth_log {
        Access-Accept = "%t : eduroam-auth#ORG=%{request:Realm}#USER=%{User-Name}#CSI=%{%{Calling-Station-Id}:-
Unknown Caller Id}#NAS=%{%{Called-Station-Id}:-Unknown Access Point}#CUI=%{%{reply:Chargeable-User-Identity}:-
Unknown}#MSG=%{%{EAP-Message}:-No EAP Message}#RESULT=OK#"
        Access-Reject = "%t : eduroam-auth#ORG=%{request:Realm}#USER=%{User-Name}#CSI=%{%{Calling-Station-Id}:-
Unknown Caller Id}#NAS=%{%{Called-Station-Id}:-Unknown Access Point}#CUI=%{%{reply:Chargeable-User-Identity}:-
Unknown}#MSG=%{%{reply:Reply-Message}:-No Failure Reason}#RESULT=FAIL#"
    }
}
```

1.1.1.8.2. Enable modules

```
cd mods-enabled; ln -s ../mods-available/eduroam_cui_log; ln -s ../mods-available/cui
```

1.1.1.8.3. Client definition

Force parameter 'add_cui' to 'yes' for all your connected clients :

```
client antarctica-access-point-1 {
...
    add_cui = yes
 }
```

1.1.1.8.4. Policy

Edit the default policy.d/cui file :

```
...
cui_hash_key = "changeme"         # --> replace with a random string
                                                        # if you use a secondary or backup FreeRADIUS server,
use the same cui_hash_key
                                                        # this allows you to keep the same CUI log even if the
FreeRADIUS server change
cui_require_operator_name = "yes"
...
```

Others values don't need to be changed.

1.1.1.8.5. Attributes

Edit mods-config/attr_filter/pre-proxy file, check that attributes Calling-Station-Id, Operator-Name and Chargeable-User-Identity are defined :

```
DEFAULT
...
        Calling-Station-Id =* ANY,
        Operator-Name =* ANY,
        Chargeable-User-Identity =* ANY,
...
```

Edit mods-config/attr_filter/post-proxy file, check that the attributes User-Name and Chargeable-User-Identity are defined :

```
DEFAULT
...
        User-Name =* ANY,
        Chargeable-User-Identity =* ANY,
...
```

1.1.1.8.6. CUI filtering

Edit policy.d/filter, add a filter function 'cui_filter'. Simple example :

```
# Filter the Chargeable-User-Identity attribute
cui_filter {
  if (&reply:Chargeable-User-Identity =~ /REPLACE-WITH-CUI-TO-MATCH/) {
                        update request {
                                &Module-Failure-Message += "Rejected: CUI matching '%{reply:Chargeable-User-
Identity}'"
                        }
                        reject
       }
}
```

1.1.1.8.7. Using policies and modules in your eduroam virtual server

Add 'cui' in authorize, post-auth and pre-proxy sections. Add 'cui_log' and 'cui_filter' in post-auth section :

```
server eduroam {
...
        authorize {
                # only use filter_username from version > 3.0.7 on
                filter_username
                update request {
                        Operator-Name := "1yourdomain.tld"
                        # the literal number "1" above is an important prefix! Do not change it!
                }
                cui
                                # if you want detailed logging
                auth_log
                suffix
        }
...
        post-auth {
                # if you want detailed logging
                reply_log
                                cui
                                cui_filter
                                cui_log
                Post-Auth-Type REJECT {
                        reply_log
                                                eduroam_log
                }
        }
...
        pre-proxy {
                pre_proxy_log
                cui
                if("%{Packet-Type}" != "Accounting-Request") {
                        attr_filter.pre-proxy
                }
        }
...
}
```

1.1.1.9. Caveats

Use the most recent version available (3.0.10 at the time of writing) because of known issues in older versions (ranging from filters that prevent people to get online with mixed usernames to TLS-related bugs).