Installation of Eduroam IRS with Freeradius on Ubuntu22.04

It is assumed that this installation will be carried on a fresh installation of ubuntu 22.04 server.

Update your server

```
sudo apt update

sudo apt upgrade
```

FreeRADIUS 3.2 on Ubuntu Jammy 22.04

Add the NetworkRADIUS PGP public key:

```
install -d -o root -g root -m 0755 /etc/apt/keyrings
curl -s 'https://packages.networkradius.com/pgp/packages%40networkradius.com' | \
    sudo tee /etc/apt/keyrings/packages.networkradius.com.asc > /dev/null
```

Add an APT preferences file to ensure all freeradius packages are installed from the Network RADIUS repository:

```
printf 'Package: /freeradius/\nPin: origin "packages.networkradius.com"\nPin-
Priority: 999\n' | \
    sudo tee /etc/apt/preferences.d/networkradius > /dev/null
```

Add the APT sources list:

```
echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/packages.networkradius.com.asc]
http://packages.networkradius.com/freeradius-3.2/ubuntu/jammy jammy main" | \
    sudo tee /etc/apt/sources.list.d/networkradius.list > /dev/null
```

Finally, update the APT database and install the packages:

```
sudo apt update
```

Install Packages

You need to become root by `sudo su` and proceed.

```
sudo apt install freeradius

sudo apt install git libssl-dev devscripts pkg-config libnl-3-dev libnl-genl-3-dev
```

Next, `sudo vim /etc/freeradius/users` and modify to enable bob and test realm user

```
#
#bob     Cleartext-Password := "hello"
#       Reply-Message := "Hello, %{User-Name}"
#
eduroamtest         Cleartext-Password := "EduTestP@33"
####
```

After the user modification following radtests should succeed.

```
sudo systemctl restart freeradius.service
radtest -t mschap -x eduroamtest  EduTestP@33 127.0.0.1:1812 10000 testing123
```

Build eapol tool

```
git clone --depth 1 --no-single-branch https://github.com/FreeRADIUS/freeradius-server.git

cd freeradius-server/scripts/ci/

./eapol_test-build.sh

sudo cp ./eapol_test/eapol_test /usr/local/bin/
```

You can now test the above radius local user authentication with eapol_test as below. For eapol_test command you need to create a configuration file which describes the network connection properties. Let's create a configuration file.

```
sudo vim peap-mschapv2-local.conf
```

Add the below code,

```
network={
        ssid="eduroam"
        key_mgmt=WPA-EAP
        eap=PEAP
        identity="eduroamtest"
#       anonymous_identity="@eduroam.lk"
        password="EduTestP@33"
        phase2="auth=MSCHAPV2"

        #  Uncomment the following to perform server certificate validation.
#       ca_cert="/etc/raddb/certs/ca.der"
}
```

And execute the eapol_test as below,

```
eapol_test -c peap-mschapv2-local.conf -p 1812 -s testing123
```

If the authentication is successful you should recieve at the end,

```
MPPE keys OK: 1  mismatch: 0
SUCCESS
```

Freeradius Settings

Go to install location and do the changes.

```
cd /etc/freeradius/
sudo cp mods-config/attr_filter/pre-proxy mods-config/attr_filter/pre-proxy.orig
sudo cp mods-config/attr_filter/post-proxy mods-config/attr_filter/post-proxy.orig
```

Edit the file `pre-proxy` with following content:

```
sudo vim mods-config/attr_filter/pre-proxy

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,
        Class =* ANY,
        Chargeable-User-Identity =* ANY
```

Edit the file `post-proxy` with following content:

```
sudo vim mods-config/attr_filter/post-proxy

DEFAULT
        Framed-IP-Address == 255.255.255.254,
        Framed-IP-Netmask == 255.255.255.255,
        Framed-MTU >= 576,
        Framed-Filter-ID =* ANY,
        Reply-Message =* ANY,
        Proxy-State =* ANY,
        EAP-Message =* ANY,
        Message-Authenticator =* ANY,
        MS-MPPE-Recv-Key =* ANY,
        MS-MPPE-Send-Key =* ANY,
        MS-CHAP-MPPE-Keys =* ANY,
        State =* ANY,
        Session-Timeout <= 28800,
        Idle-Timeout <= 600,
        Calling-Station-Id =* ANY,
        Operator-Name =* ANY,
        Port-Limit <= 2,
        User-Name =* ANY,
        Class =* ANY,
        Chargeable-User-Identity =* ANY
```

Backup the eap module configuration file as follows,

```
sudo cp mods-available/eap mods-available/eap.orig

sudo vim mods-enabled/eap
```

Now modify the configuration file to make the below changes. Don't delete any additional configurations not show below. Also some of the below configurations also might be the same as them in your configuration file, hence need to change the selected parts only.

```
eap {
                default_eap_type = peap     # change to your organisation's
preferred eap type (tls, ttls, peap, mschapv2)
                timer_expire     = 60
                ignore_unknown_eap_types = no
```

```
                cisco_accounting_username_bug = no

                tls-config tls-eduroam {
                        private_key_password = whatever
                        private_key_file = ${certdir}/server.pem
                        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"
                }

                tls {
                        tls = tls-eduroam
                }

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

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

                mschapv2 {
                #       send_error = yes
                }

        }
```

You need to modify the linelog module as follows too,

```
sudo vim mods-enabled/linelog
```

Modify the following lines containing `Access-Accept` and `Access-Reject`

```
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}#NAS-IP=%{%{NAS-IP-Address}:-Unknown}#OPERATOR=%{%{Operator-
Name}:-Unknown}#CUI=%{%{reply:Chargeable-User-Identity}:-Unknown}#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}#NAS-IP=%{%{NAS-IP-Address}:-Unknown}#OPERATOR=%{%{Operator-
Name}:-Unknown}#CUI=%{%{reply:Chargeable-User-Identity}:-
Unknown}#MSG=%{%{reply:Reply-Message}:-No Failure Reason}#RESULT=FAIL#"
```

Chargeable-User-Identity (CUI) is a non-human readable (“opaque”) cryptographic hash that is targeted to the service provider. Each service provider therefore receives

a different opaque value for the same user. This allows service providers to recognize a user as one that they have seen before, without knowing who the user is; while preventing service providers from colluding to track users. This enables legitimate purposes, such as blocking malfunctioning devices and generating accurate usage statistics. The CUI value is computed as a SHA1 hash of concatenated (inner) User-Name, optional Operator-Name and a local salt value. This salt is random string and we have to set this salt in the cui_hash_key attribute.

Modify the cui policy as follows,

```
sudo vim policy.d/cui

cui_hash_key = "SOMELONGCHARACTERstring"
cui_require_operator_name = "yes"
```

Creating Certificates

Certificates can be obtained using a service like LetsEncrypt or Commercial provider. We can also create certificates using a private CA.You need to only follow a one method.

Create certificates using LetsEncrypt

```
apt-get install certbot
addgroup certs
adduser freerad certs

certbot certonly  --standalone  --cert-name SERVER_FQDN  -d SERVER_FQDN
```

certificates will be created at /etc/letsencrypt/live/SERVER_FQDN/. Server certificate along with CA certificates will be in a file named fullchain.pem and private key will be in privkey.pem.

Now you need to edit eap module configuration file and replace the lines below as given.

```
nano mods-enabled/eap

private_key_file = /etc/letsencrypt/live/SERVER_FQDN/privkey.pem
certificate_file = /etc/letsencrypt/live/SERVER_FQDN/fullchain.pem
```

Create Certificates Using Private CA

```
cd /etc/freeradius/certs/
```

edit `[certificate_authority]` of `/etc/freeradius/certs/ca.cnf` similar to the below. Make changes to reflect your institute.

```
countryName             = LK
stateOrProvinceName     = Central
localityName            = Somewhere
organizationName        = Univerity of ABC
```

```
emailAddress            = admin@YOUR_DOMAIN
commonName              = "Univerity of ABC Certificate Authority"
```

edit `[server]` of `/etc/freeradius/certs/server.cnf` similar to the below as well. Make changes to reflect your institute.

```
[server]
countryName             = LK
stateOrProvinceName     = Central
localityName            = Somewhere
organizationName        = Univerity of ABC
emailAddress            = irs.admin@YOUR_DOMAIN
commonName              = "irs.YOUR_DOMAIN"
```

Then build the certificates,

```
cd /etc/freeradius/certs
make ca.pem
make server.pem
chown freerad:freerad *
```

Create virtual server for eduroam as

```
cd /etc/freeradius/
sudo vim sites-available/eduroam

##############################################################################
#
# Virtual Server Eduroam
#
##############################################################################


server eduroam {

listen {
        type = auth
        ipaddr = *
        port = 0
        limit {
              max_connections = 16
              lifetime = 0
              idle_timeout = 30
        }
}

listen {
        ipaddr = *
        port = 0
        type = acct
        limit {
        }
}


listen {
        type = auth
        ipv6addr = ::
        port = 0
```

```
        limit {
              max_connections = 16
              lifetime = 0
              idle_timeout = 30
        }
}

listen {
        ipv6addr = ::
        port = 0
        type = acct
        limit {

        }
}


authorize {

        preprocess
        filter_username
       if (("%{client:shortname}" != "FLR1")||("%{client:shortname}" != "FLR2"))
{
                  update request {
                         Operator-Name := "1YOUR_DOMAIN"
                          # the literal number "1" above is an important prefix!
Do not change it!
         }
        }
        operator-name
        cui
        auth_log
        suffix
        eap {
                ok = return
        }
        files

        -ldap

}



authenticate {

        eap


}


preacct {
        suffix

}


accounting {
```

```
}

session {
}


post-auth {

        update {
                &reply: += &session-state:
        }

        reply_log
        linelog
        remove_reply_message_if_eap
        Post-Auth-Type REJECT {
                reply_log
                linelog
        }
}


pre-proxy {

        # if you want detailed logging
        cui
        pre_proxy_log  # logs the packet to the file system again. Attributes
that have been added on during inspection are now visible

        if("%{Packet-Type}" != "Accounting-Request") {

              attr_filter.pre-proxy   # removes unnecessary attributes off of the
request before sending the request upstream

                }
}

post-proxy {
                # if you want detailed logging

                post_proxy_log              # logs the rply 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)


}
}
```

Create virtual server for eduroam-inner-tunnel.

```
sudo vim sites-available/eduroam-inner-tunnel

##############################################################################
#
```

```
# Virtual Server Eduroam-Inner-Tunnel
#
##########################################################################
server eduroam-inner-tunnel {

listen {
       ipaddr = 127.0.0.1
       port = 18120
       type = auth
}

authorize {
        auth_log
        suffix
        update control {
                        &Proxy-To-Realm := LOCAL
        }
        eap {
                ok = return
        }
        files
        -ldap
        mschap
        pap
}

authenticate {

        Auth-Type PAP {
                pap
        }

        Auth-Type MS-CHAP {
                mschap
        }

        eap
}


session {
        radutmp

}



post-auth {
        cui-inner
        reply_log
        Post-Auth-Type REJECT {
                reply_log
                attr_filter.access_reject
                update outer.session-state {
                        &Module-Failure-Message := &request:Module-Failure-
Message
                }
        }
}
```

```
pre-proxy {

}


post-proxy {
        eap
}
}
```

Create `vim sites-available/blackhole` for blackholing

```
server blackhole {
    authorize {
        reject
    }
}
```

Next you need to enable the created virtual-server sites above and also remove the unwanted,

```
cd sites-enabled
rm default
rm inner-tunnel
ln -s ../sites-available/eduroam-inner-tunnel eduroam-inner-tunnel
ln -s ../sites-available/eduroam eduroam
ln -s ../sites-available/blackhole blackhole
```

Then modify proxy.conf

```
cd /etc/freeradius
sudo cp proxy.conf proxy.conf.orig
sudo vim proxy.conf

proxy server {
        default_fallback        = no
}

# Add your country's FLR details for the home_server {} attribute as shown below.
port and status_check will not change.
# Add as many definitions as there are FLRs
# nro1.learn.ac.lk and nro2.learn.ac.lk are for Sri Lanka maintained by LEARN.
home_server FLR1 {
        ipaddr                  = nrs1.ac.lk
        port                    = 1812
        secret                  = FLR_EDUROAM_SECRET
        status_check            = status-server
}
home_server FLR2 {
        ipaddr                  = nrs2.ac.lk
        port                    = 1812
        secret                  = FLR_EDUROAM_SECRET
        status_check            = status-server
}

realm LOCAL {
        #  If we do not specify a server pool, the realm is LOCAL, and
```

```
        #  requests are not proxied to it.
}

realm NULL {
        # If a user types their username without the domain, it will end up here
}


# eduroam home_server_pool attribute links from the home_server attribute. ensure
home_server in home_server_pool matches home_server above
home_server_pool EDUROAM {
        type                    = fail-over
        home_server             = FLR1
        home_server              = FLR2
}


# Your IdP realm
realm YOUR_DOMAIN {
       # nostrip #uncomment to remove striping of realm from username
}

# Catchall for unhandled realms
# redirect them to a blackhole server
#
home_server blackhole {
    virtual_server = blackhole
}

home_server_pool blackhole_pool {
    home_server = blackhole
    name = blackhole
}

realm  wlan.mnc000.mcc413.3gppnetwork.org{
    auth_pool = blackhole_pool
}

realm  wlan.mnc001.mcc413.3gppnetwork.org{
    auth_pool = blackhole_pool
}

realm  wlan.mnc002.mcc413.3gppnetwork.org{
    auth_pool = blackhole_pool
}

realm  wlan.mnc003.mcc413.3gppnetwork.org{
auth_pool = blackhole_pool
}

realm  wlan.mnc004.mcc413.3gppnetwork.org{
    auth_pool = blackhole_pool
}

realm  wlan.mnc005.mcc413.3gppnetwork.org{
    auth_pool = blackhole_pool
}

realm  wlan.mnc006.mcc413.3gppnetwork.org{
```

```
auth_pool = blackhole_pool
}

realm  wlan.mnc007.mcc413.3gppnetwork.org{
    auth_pool = blackhole_pool
}

realm  wlan.mnc008.mcc413.3gppnetwork.org{
    auth_pool = blackhole_pool
}

realm  wlan.mnc009.mcc413.3gppnetwork.org{
auth_pool = blackhole_pool
}

#########################################
# Proxy the rest
realm "~.+$" {
        pool                    = EDUROAM
        nostrip
}
```

```
#####################################################
```

Modify Clients

```
vi clients.conf
```

Add following to the tail

```
client FLR1 {
        ipaddr          = nrs1.ac.lk
        secret          = FLR_EDUROAM_SECRET
        shortname       = FLR1
        nas_type  = other
        Operator-Name = 1YOUR_DOMAIN
        add_cui = yes
    virtual_server = eduroam
}


client FLR2 {
        ipaddr           = nrs2.ac.lk
        secret          = FLR_EDUROAM_SECRET
        shortname       = FLR2
        nas_type        = other
        Operator-Name = 1YOUR_DOMAIN
        add_cui = yes
    virtual_server = eduroam
}
```

Now you should contact your National Roaming Operator and get your shared keys.

You may also need to add all clients directly connecting to the radius, such as AP's and controllers... To add an Aruba access points add something like below.

```
client aruba_aps {
        ipaddr = 192.248.4.224/27
        secret = ArubaAPSECRET
        Operator-Name = 1YOUR_DOMAIN
        add_cui = yes
        limit {
          max_connections = 10
        }

}
```

Now restart the radius server.

```
sudo systemctl restart freeradius.service
```

After the restart, following tests should succeed.

```
eapol_test -c peap-mschapv2-local.conf -p 1812 -s testing123
```

You may also test some of the test roaming accounts provided by your upstream NRO.

Enabling LDAP users

Install Freeradius LDAP module

```
apt-get install freeradius-ldap
```

Configure LDAP parameters

```
sudo vim /etc/freeradius/mods-available/ldap
```

Add or Modify the appopriate lines

```
server = 'LDAP_SERVER_FQDN'
identity = 'cn=admin,dc=inst,dc=ac,dc=lk' #bind User
password = 'YOUR_LDAP_PASSWORD'
base_dn = 'ou=people,dc=inst,dc=ac,dc=lk'
edir_autz = yes
```

(You should consider connecting LDAP with STARTTLS enable. Please consult the ldap module for configurations)

Enable LDAP Module & Restart Freeradius

```
ln -s /etc/freeradius/mods-available/ldap /etc/freeradius/mods-enabled/ldap
service freeradius restart
```

Network configuration file for the ldap connectivity may look like below.

```
network={
        ssid="eduroam"
        key_mgmt=WPA-EAP
        eap=PEAP
        identity="user@YOUR_DOMAIN"
#       anonymous_identity="@eduroam.lk"
```

```
        password="USER-PASSWORD"
        phase2="auth=MSCHAPV2"
        #  Uncomment the following to perform server certificate validation.
#       ca_cert="/etc/raddb/certs/ca.der"
}
```

Test ldap user authentication:

```
eapol_test -c peap-mschapv2-ldap.conf -p 1812 -s testing123
```

Troubleshoot:

Log Path: `/var/logs/freeradius/`
Debug mode:

- In a new console, stop freeradius service `service freeradius stop`
- Start in debug mode `freeradius -X`
- To stop debug mode, use CTRL+c