eduroam with LDAP Authentication

Note: FreeRADIUS Directory Structure

/etc/freeradius/3.0/clients.conf – RADIUS clients (APs, switches)

/etc/freeradius/3.0/users – Local test users

/etc/freeradius/3.0/mods-enabled/eap – EAP configuration

/etc/freeradius/3.0/sites-enabled/default – Authentication logic

Important: Replace on vmxx replace xx with your virtual machine number

Install FreeRADIUS along with LDAP modules and verify a working FreeRADIUS service.

```
sudo apt update
```

```
sudo apt install freeradius freeradius-ldap -y
```

Activating and starting FreeRADIUS:

```
sudo systemctl enable freeradius
```

```
freeradius -X
```

Expected Result:

- FreeRADIUS starts without errors

```
 sudo systemctl start freeradius
sudo systemctl status freeradius
```

Configure FreeRADIUS to authenticate users from LDAP.

step 1: Configure the LDAP Module

```
nano /etc/freeradius/3.0/mods-available/ldap
```

```
ldap {
        server = 'vmxx.kenet.ac.ke'
        identity = 'cn=admin,dc=vmxx,dc=kenet,dc=ac,dc=ke'
        password = 'StrongPassword'
        base_dn = 'dc=vmxx,dc=kenet,dc=ac,dc=ke'
```

Enable LDAP Module

```
ln -s /etc/freeradius/3.0/mods-available/ldap /etc/freeradius/3.0/mods-enabled/
```

Now change the group rights of the file we just edited:

```
sudo chgrp -h freerad /etc/freeradius/3.0/mods-available/ldap
sudo chown -R freerad:freerad /etc/freeradius/3.0/mods-enabled/ldap
```

And restart the FreeRADIUS service:

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

---

step 2: Configure eduroam inner tunnel to use LDAP authentication

```
nano  /etc/freeradius/3.0/sites-enabled/eduroam-inner-tunnel
```

```
server eduroam-inner-tunnel {
authorize {
        filter_username
        ldap
        suffix
        update control {
                &Proxy-To-Realm := LOCAL
        }
        eap {
                ok = return
        }
        expiration
        logintime
        pap
        if (!&control:Auth-Type && &User-Password) {
                update control {
                       &Auth-Type := LDAP
                }
        }
        chap
        mschap
}
authenticate {
        Auth-Type PAP {
                pap
        }
        Auth-Type CHAP {
                chap
        }
        Auth-Type LDAP {
                ldap
        }
        eap
        Auth-Type MS-CHAP {
                mschap
        }
}
session {
        radutmp
}
post-auth {
        reply_log
        -sql
        ldap
        if (1) {
                update reply {
                        &User-Name !* ANY
                        &Message-Authenticator !* ANY
                        &EAP-Message !* ANY
                        &Proxy-State !* ANY
                        &MS-MPPE-Encryption-Types !* ANY
                        &MS-MPPE-Encryption-Policy !* ANY
                        &MS-MPPE-Send-Key !* ANY
                        &MS-MPPE-Recv-Key !* ANY
                }
                update {
                        &outer.session-state: += &reply:
                }
        }
        Post-Auth-Type REJECT {
                -sql
                attr_filter.access_reject
                update outer.session-state {
                        &Module-Failure-Message := &request:Module-Failure-Message
                }
        }
}
pre-proxy {
}
post-proxy {
        eap
}
} # inner-tunnel server block
```

---

step 3: Configure eduroam to use LDAP authentication

Finally, let’s make sure we have ldap authentication configured in our eduroam site as well.

```
nano /etc/freeradius/3.0/sites-enabled/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 = ::   # any.  ::1 == localhost
        port = 0
        limit {
              max_connections = 16
              lifetime = 0
              idle_timeout = 30
```

```
		}
	}
	listen {
		ipv6addr = ::
		port = 0
		type = acct
		limit {
		}
	}
	authorize {
		filter_username
		preprocess
		auth_log
		digest
		suffix
		eap {
			ok = return
		}
		ldap
		if ((ok || updated) && User-Password && !control:Auth-Type) {
			update control {
				&Auth-Type := ldap
			}
		}
		expiration
		logintime
		pap
		Autz-Type New-TLS-Connection {
			  ok
		}
		chap
		mschap
	}
	authenticate {
		Auth-Type PAP {
			pap
		}
		Auth-Type CHAP {
			chap
		}
		digest
		Auth-Type LDAP {
			ldap
		}
		eap
		Auth-Type eap {
			eap {
```

```
                          handled = 1
                  }
                  if (handled && (Response-Packet-Type == Access-Challenge)) {
                          attr_filter.access_challenge.post-auth
                          handled  # override the "updated" code from attr_filter
                  }
          }
          Auth-Type MS-CHAP {
                  mschap
          }
  }
  preacct {
          preprocess
          acct_unique
          suffix
          files
  }
  accounting {
          detail
          unix
          -sql
          exec
          attr_filter.accounting_response
  }
  session {
  }
  post-auth {
          if (!&reply:State) {
                  update reply {
                          State := "0x%{randstr:16h}"
                  }
          }
          verify_tls_client_common_name
          if (&EAP-Message && !&Stripped-User-Name && &TLS-Client-Cert-Serial) {
                  update request {
                          &Stripped-User-Name := "%{%{TLS-Client-Cert-Subject-Alt-Name-Email}:-%{%{TLS-C
                  }
                  update reply {
                          Class += "%{md5:%{Calling-Station-Id}%{Called-Station-Id}%{TLS-Client-Cert-Sub
                  }
          }
          if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name == req
                  update reply {
                          &User-Name !* ANY
                  }
          }
          update {
                &reply: += &session-state:
        }
        if (!Realm) {
         update reply {
            Realm := "%{substring:%{User-Name}:@}"
                }
        }
        reply_log
        -sql
        ldap
        exec
        if (&reply:EAP-Session-Id) {
                update reply {
                        EAP-Key-Name := &reply:EAP-Session-Id
                }
        }
        remove_reply_message_if_eap
        Post-Auth-Type REJECT {
                -sql
                attr_filter.access_reject
                eap
                remove_reply_message_if_eap
                linelog
        }
        Post-Auth-Type Challenge {
        }
        Post-Auth-Type Client-Lost {
        }
        if (EAP-Key-Name && &reply:EAP-Session-Id) {
                update reply {
                        &EAP-Key-Name := &reply:EAP-Session-Id
                }
        }
        linelog
}
pre-proxy {
}
post-proxy {
        eap
}
}
```

Recommended Method

- EAP-TTLS

- Inner method: PAP

  Modify EAP module as below

```
nano /etc/freeradius/3.0/mods-enabled/eap
```

```
eap {
        default_eap_type = ttls
        timer_expire = 60
        ignore_unknown_eap_types = no
        cisco_accounting_username_bug = no
        max_sessions = ${max_requests}
        md5 {
        }
        gtc {
                auth_type = LDAP
        }
        tls-config tls-common {
private_key_password = whatever
                private_key_file = /etc/ssl/private/ssl-cert-snakeoil.key
                certificate_file = /etc/ssl/certs/ssl-cert-snakeoil.pem
                ca_file = /etc/ssl/certs/ca-certificates.crt

                 ca_file = /etc/ssl/certs/ca-certificates.crt
                ca_path = ${cadir}
                cipher_list = "DEFAULT"
                cipher_server_preference = no
                tls_min_version = "1.2"
                tls_max_version = "1.2"
                ecdh_curve = ""
                cache {
                        enable = yes
                        lifetime = 24 # hours
                        store {
                                Tunnel-Private-Group-Id
                        }
                }
                verify {
                }
                ocsp {
                        enable = no
                        override_cert_url = yes
                        url = "http://127.0.0.1/ocsp/"
                }
        }
        tls {
                tls = tls-common
        }
 ttls {
                tls = tls-common
                default_eap_type = gtc
                copy_request_to_tunnel = yes
                use_tunneled_reply = yes
                virtual_server = "eduroam-inner-tunnel"
        }
        peap {
                tls = tls-common
                default_eap_type = mschapv2
                copy_request_to_tunnel = no
                use_tunneled_reply = no
                virtual_server = "eduroam-inner-tunnel"
        }
        mschapv2 {
        }
 }
```

Configure Local Realm

```
nano /etc/freeradius/3.0/proxy.conf
```

Add

```
realm vmxx.kenet.ac.ke {
       auth_pool = my_auth_failover
 nostrip
}
```

Make sure ldap server already added on client configuration
Replace x.x.x.x with the IP of the LDAP server

```
nano /etc/freeradius/3.0/clients.conf
```

```
client LDAP {
      ipaddr = x.x.x.x
      secret = StrongSecret
}
```

Testing with radtest

Use radtest to verify user authentication:

```
radtest test_user your_password localhost 0 my_shared_secret
```

Replace test_user, your_password, and my_shared_secret with your configured values.

```
 radtest mcurie 2QEq8qaChemists localhost 0 testing123
radtest mcurie 2QEq8qaChemists x.x.x.x 0 StrongSecret
radtest mcurie 2QEq8qaChemists x.x.x.x 1812 StrongSecret
```

Observe Logs

```
freeradius -X
```

Expected Result:

- Access-Accept returned

CONFIGURE SSL CERTIFICATES

This directory contains scripts to create the server certificates.
/etc/freeradius/3.0/certs/
Change to the certs directory

```
cd /etc/freeradius/3.0/certs/
```

Create a backup of the certs dir

```
 mkdir /etc/freeradius/3.0/certsbak
cp -prv /etc/freeradius/3.0/certs/* /etc/freeradius/3.0/certsbak/
```

NEW INSTALLATIONS OF FREERADIUS

```
cd /etc/freeradius/3.0/certs/
```

The old test certificates can be deleted by
running the following command:

```
rm -f *.pem *.der *.csr *.crt *.key *.p12 serial* index.txt*
```

Then, follow the instructions below for creating real certificates. Once the final certificates have been created, you can delete the "bootstrap" command from this directory, and delete the "make_cert_command" configuration from the "tls" sub-section of
eap.conf.

step 1: Making a Root Certificate

Edit the "input_password" and "output_password" fields to be the
password for the CA certificate.

Edit the [certificate_authority] section to have the correct values
for your country, state, etc.

```
vi ca.cnf
```

---

default_days = 3650
default_crl_days = 365
aTV3V0w
[req]
prompt = no
distinguished_name = certificate_authority
default_bits = 2048
input_password = Yourcertpassword
output_password = Yourcertpassword
x509_extensions = v3_ca

[certificate_authority]
countryName = KE
stateOrProvinceName = Nairobi

localityName = Nairobi

organizationName = Institution University

emailAddress = info@vmxx.kenet.ac.ke

commonName = "Institution University Eduroam Certificate Authority"

```
make ca.pem
```

This step creates the CA certificate.

```
make ca.der
```

This step creates the DER format of the self-signed certificate, which is can be imported into Windows.

step 2: Making a Server Certificate

The following steps will let you create a server certificate for use
with TLS-based EAP methods, such as EAP-TLS, PEAP, and TTLS. Follow
similar steps to create an "inner-server.pem" file, for use with
EAP-TLS that is tunneled inside of another TLS-based EAP method.

```
vi server.cnf
```

Edit the "input_password" and "output_password" fields to be the
password for the server certificate.

Edit the [server] section to have the correct values for your country, state, etc. Be sure that the commonName field here is different from the commonName for the CA certificate.

default_days = 3650
default_crl_days = 365

[req]
prompt = no
distinguished_name = server
default_bits = 2048
input_password = Yourcertpassword
output_password = Yourcertpassword

[server]

countryName = KE

stateOrProvinceName = Nairobi

localityName = Nairobi

organizationName = Institution University

emailAddress = info@vmxx.kenet.ac.ke

commonName = vmxx.kenet.ac.ke

```
make server.pem
```

This step creates the server certificate.
If you have an existing certificate authority, and wish to create a certificate signing request for the server certificate, edit server.cnf as above, and type the following command.

```
make server.csr
```

You will have to ensure that the certificate contains the XP extensions needed by Microsoft clients.

---

step 3: Making a Client certificate

Client certificates are used by EAP-TLS, and optionally by EAP-TTLS and PEAP. The following steps outline how to create a client certificate that is signed by the server certificate created above.
You will have to have the password for the server certificate in the "input_password" and "output_password" fields of the server.cnf file.

```
vi client.cnf
```

Edit the "input_password" and "output_password" fields to be the password for the client certificate. You will have to give these passwords to the end user who will be using the certificates.

Edit the [client] section to have the correct values for your country, state, etc. Be sure that the commonName field here is the User-Name that will be used for logins!

default_days = 3650
default_crl_days = 365

[req]
prompt = no
distinguished_name = client
default_bits = 2048
input_password = Yourcertpassword
output_password = Yourcertpassword

[client]
countryName = KE
stateOrProvinceName = Nairobi
localityName = Nairobi
organizationName = Institution University
emailAddress = info@vmxx.kenet.ac.ke
commonName = "Institution University Eduroam Client Certificate"

```
make client.pem
```

The users certificate will be in "emailAddress.pem", i.e. "user@vmxx.kenet.ac.ke.pem".

step 4: Configure EAP Module

Modify EAP Module as below

```
nano /etc/freeradius/3.0/mods-enabled/eap
```

```
  tls-config tls-common {
                private_key_password = Yourcertpassword
                private_key_file = /etc/freeradius/3.0/certs/server.key
                certificate_file = /etc/freeradius/3.0/certs/server.pem
                ca_file = /etc/freeradius/3.0/certs/ca.pem
```

Change the ownership of the certificates to freerad user

```
chown freerad:freerad  client.pem server.pem ca.pem ca.der server.key
```

To check validity of certs

```
openssl x509 -text -noout -in /etc/freeradius/3.0/certs/ca.pem
```

Restart the Freeradius service and check the status

```
 sudo systemctl start freeradius
```

```
sudo systemctl status freeradius
```

- For SP role (accepting visitors): Proxy non-local realms to your national proxy.
- For IdP role (authenticating own users): Handle your own realm locally and proxy others if needed.
- In FreeRADIUS, this is done in `proxy.conf` :

```
 home_server eduroam_proxy {
     type = auth+acct
     ipaddr = roaming1.eduroam.ac.ke   # Your national proxy IP (e.g., KENET's)
     port = 1812
     secret = follow-your-NRO-procedure
     response_check = "noop"
}

home_server_pool eduroam_pool {
     type = fail-over
```

```
        home_server = eduroam_proxy
    }

    realm DEFAULT {
        pool = eduroam_pool
    }

    realm YOUR.REALM.AC.KE {
        # Handle locally (no proxy)
    }
```

- **Important**: Use `realm DEFAULT` for proxying everything else. Many NROs provide exact IPs, secrets, and RadSec configs.