==

Steps to Run MUD Controller and FreeRadius

==

INITIAL NOTE: ***The following test steps are tested on Ubuntu machine and this method assumes that freeradius is already installed on user machine. Please check www.freeradius.org to install freeradius.***

AT ENTERPRISE SIDE WHERE MUD FILES ARE STORED

1. **Create a new mud directory in your workspace**

   ```
   $ mkdir mud
   ```

   Follow this link to generate MUD file
   https://www.ofcourseimright.com/mudmaker/

   # Once you specified your options and details, click on submit button to generate the MUD file

   # Copy the mud string and paste it on new mud file named with **.json** extension and save it in mud directory

2. **Generate the cms signature file using following command in the mud folder**

   ```
   $ openssl cms -sign -signer <yourfilename.json> -inkey mankey \
             -in mudfile -binary -outform DER - \
             -certfile intermediatecert -out <yourfilename.p7s>
   ```

   # For verification run below command

   ```
   $ openssl cms -verify -in <yourfilename.p7s>  -out mud.json -CAfile ca_root.pem  -inform DER –content <yourfilename.json>
   ```

   Output will show "**Verification successful**"

   # To access the mud file and signature file remotely use **SimpleHTTPServer**, as this is the simple method to access MUD files from remote server via HTTP link

   # **Run SimpleHTTPServer in your workspace directory**

   % **Note:** MUD Controller will look for the path /mud/ where MUD files exist.

   # Either run HTTP server, it runs on default port [8000]

   ```
   $ python -m SimpleHTTPServer
   ```

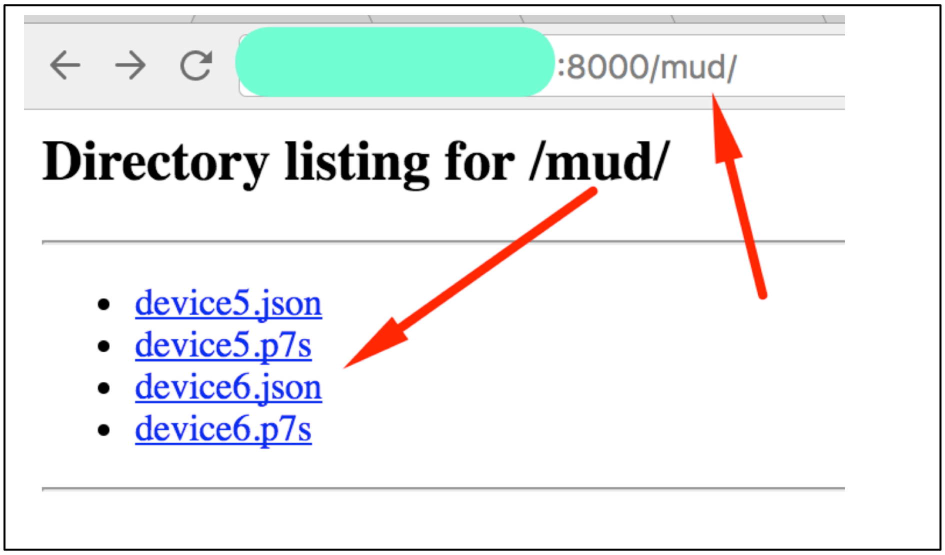
   # or to specify your IP and port

   ```
   $ python -m SimpleHTTPServer 127.0.0.1:8080
   ```

# **Open a following browser page:**

http://127.0.0.1:8000/mud/

You should see the your mud file and signature files are listed on [mud] directory which is shown on web-browser



[redacted]:8000/mud/

Directory listing for /mud/

- device5.json
- device5.p7s
- device6.json
- device6.p7s

ON FREERADIUS SERVER SIDE

1. Create controller directory at */usr/local/etc/*
   *Download and copy the **mud_controller.py** file at /usr/local/etc/**controller** directory.*

2. Created a new vendor dictionary file called ***dictionary.mudserver***, in */usr/local/etc/share/freeradius/*
   The vendor directory consist of following attributes
   --> Cisco-MUD-URI attribute used to send the MUD URI from RADCLIENT or DHCP
   --> The number 16122 is used for vendor specific attribute and it is not yet registered

Copy below text and paste in **dictionary.mudserver**

```
######################################################################
########
VENDOR              CISCO-IOT                     16122

BEGIN-VENDOR      CISCO-IOT

ATTRIBUTE     Cisco-MUD-URI                          1     string

END-VENDOR     CISCO-IOT

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

3. Open */usr/local/etc/share/freeradius/**dictionary*** file.

   Locate the lines
   $INCLUDE dictionary.motorola
   $INCLUDE dictionary.motorola.wimax

   add the following on next line
   $INCLUDE dictionary.mudserver

4. **Creating the user**

   ⇒ Add User-Name called <username> and "Cleartext-Password <password>in */usr/local/etc/raddb/users* file on **freeradius** server at staring of the line

   “<username>” Cleartext-Password “<password>”

   ⇒ Add your clients at clients.conf file in */usr/local/etc/raddb/clients.conf*

```
client <client_ipaddress> {
     ipaddr = <client_ipaddress>
```

```
                                    secret = <your password>
                                }
```

⇒ Change the **exec** configuration file in */usr/local/etc/raddb/mods-enabled/exec*
From **wait** no to yes

⇒ In */usr/local/etc/raddb/sites-enabled/default* add the following code

In “*authorize”* section add below code after “*filter_username”*

```
if (User-Name  == "%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py 'null' 'U1'}.in") {
      update control {
            Auth-Type := Accept
            }
    }
    if (User-Name  == "%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py 'null' 'U2'}.out") {
      update control {
            Auth-Type := Accept
            }
    }
```

In the same file at the “*post-auth*” section add below code after “*exec*”

```
if (Cisco-MUD-URI) {

    if (User-Name  == "<username>") {
    update reply {
                Exec-Program = "%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py %{Cisco-
MUD-URI} 'W'}"
                Cisco-AVPair := "ACS:CiscoSecure-Defined-ACL=%{exec:/usr/bin/python
/usr/local/etc/controller/mud_controller.py 'null' 'U1'}.in",
                Cisco-AVPair += "ACS:CiscoSecure-Defined-ACL=%{exec:/usr/bin/python
/usr/local/etc/controller/mud_controller.py 'null' 'U2'}.out"
                }
    }

    if (User-Name  == "%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py 'null' 'U1'}.in") {
    update reply {
                 User-Name  = "%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py 'null' 'U1'}",
                 Cisco-AVPair := "ip:inacl#1=%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py
'null' 'R1'}",
                 Cisco-AVPair += "ip:inacl#2=permit udp any any eq 67",
                 Cisco-AVPair += "ip:inacl#3=permit udp any any eq 68",
                 Cisco-AVPair += "ip:inacl#4=%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py
'null' 'R3'}",
                 Reply-Message += "DACL Ingress Downloaded Succesfully.",
                }
    }

   if (User-Name  == "%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py 'null' 'U2'}.out") {
   update reply {
                 User-Name  = "%{exec:/usr/bin/python  /usr/local/etc/controller/mud_controller.py 'null'
'U2'}.out",
                 Cisco-AVPair := "ip:outacl#1=%{exec:/usr/bin/python
/usr/local/etc/controller/mud_controller.py 'null' 'R2'}",
                 Cisco-AVPair += "ip:outacl#2=permit udp any any eq 67",
                 Cisco-AVPair += "ip:outacl#3=permit udp any any eq 68",
                 Cisco-AVPair += "ip:outacl#4=%{exec:/usr/bin/python
/usr/local/etc/controller/mud_controller.py 'null' 'R3'}",
                 Reply-Message += "DACL Egress Downloaded Succesfully."
                }
    }
```

Note: For DHCP use MAC address of the device as username

5. If using 802.1X certificate following freeradius changes are required or using only DHCP skip to next section to **Test MUD controller**

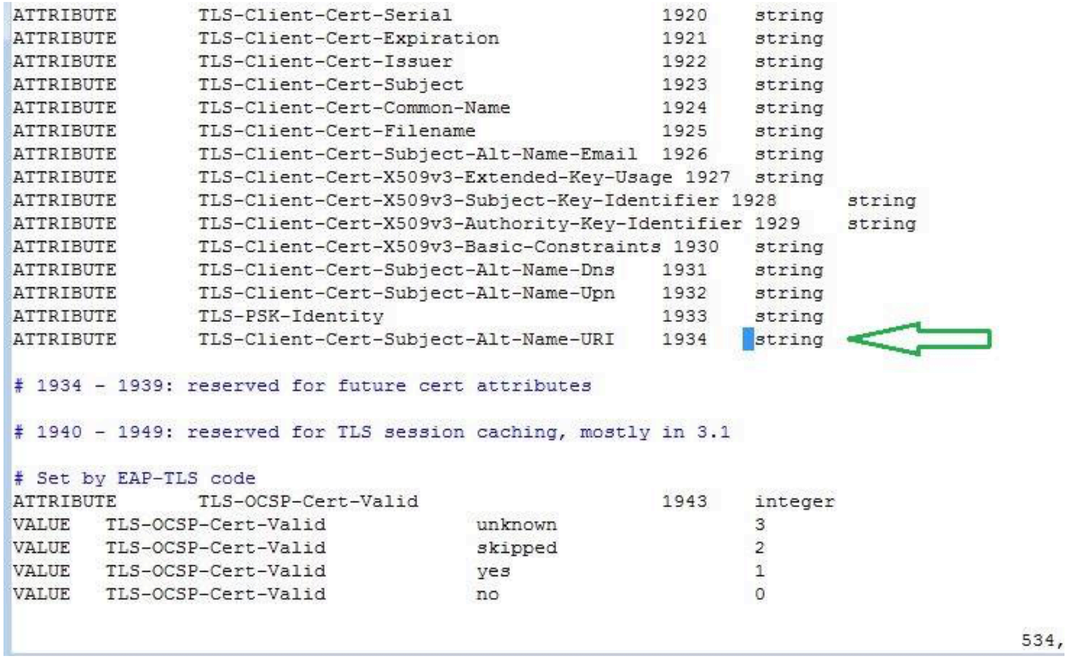
1. Download FreeRADIUS 3.0.x Series – Stable using wget
   $ wget https://github.com/FreeRADIUS/freeradius-server/archive/release_3_0_11.tar.gz

2. untar *release_3_0_11.tar.gz*
   $ *cd* ***freeradius-server-release_3_0_11/***

3. Patch the tls.c file with tls.patch in freeradius-server-release_3_0_11/src/main/
   $ *patch tls.c < tls.patch*

   In directory freeradius-server-release_3_0_11/share/ open *dictionary.freeradius.internal*
   Add following attribute after ATTRIBUTE 1933 as shown in picture below

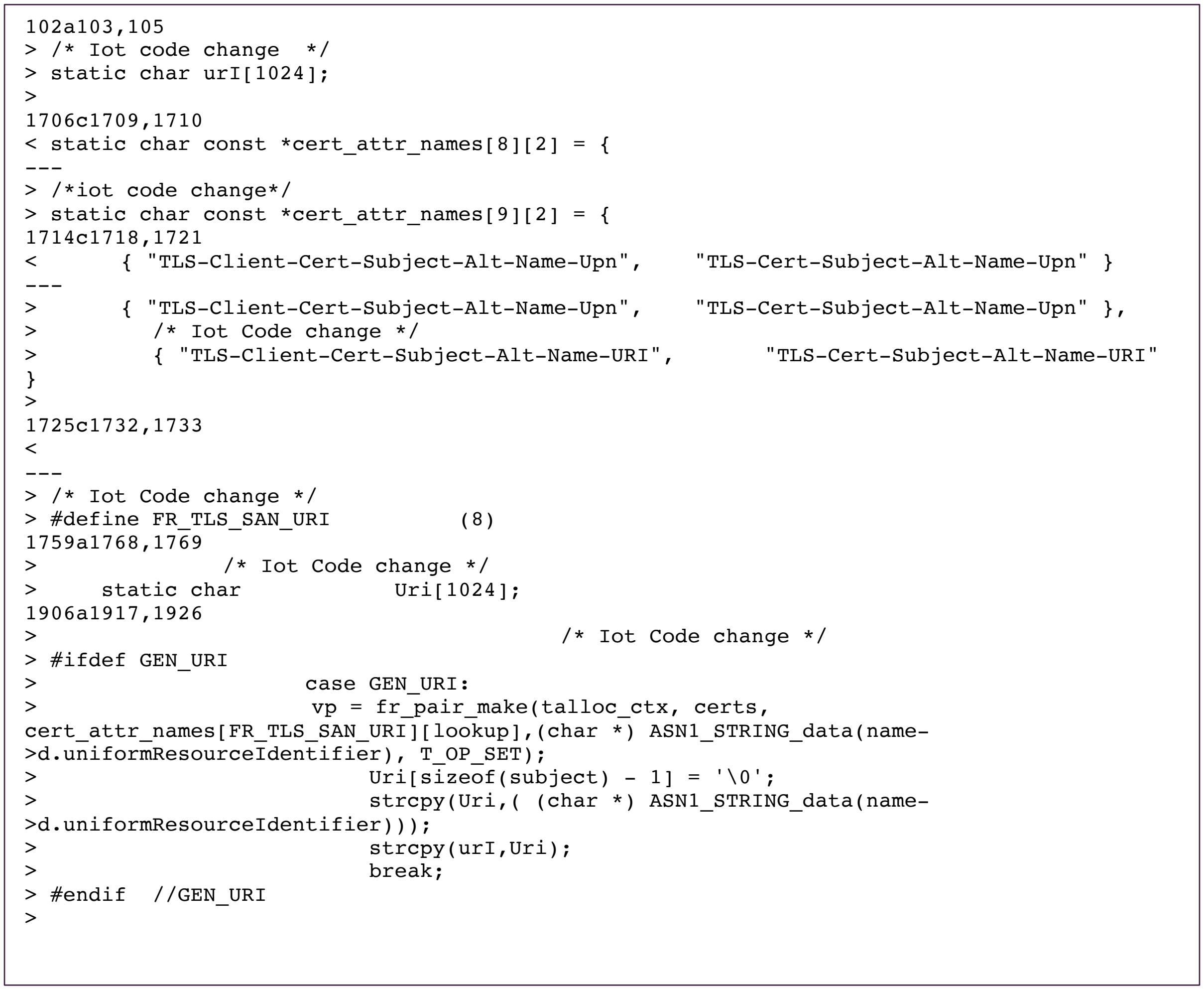
   ATTRIBUTE   TLS-Client-Cert-Subject-Alt-Name-URI   1934   string



```
ATTRIBUTE       TLS-Client-Cert-Serial                  1920    string
ATTRIBUTE       TLS-Client-Cert-Expiration              1921    string
ATTRIBUTE       TLS-Client-Cert-Issuer                  1922    string
ATTRIBUTE       TLS-Client-Cert-Subject                 1923    string
ATTRIBUTE       TLS-Client-Cert-Common-Name             1924    string
ATTRIBUTE       TLS-Client-Cert-Filename                1925    string
ATTRIBUTE       TLS-Client-Cert-Subject-Alt-Name-Email  1926    string
ATTRIBUTE       TLS-Client-Cert-X509v3-Extended-Key-Usage 1927  string
ATTRIBUTE       TLS-Client-Cert-X509v3-Subject-Key-Identifier 1928      string
ATTRIBUTE       TLS-Client-Cert-X509v3-Authority-Key-Identifier 1929    string
ATTRIBUTE       TLS-Client-Cert-X509v3-Basic-Constraints 1930   string
ATTRIBUTE       TLS-Client-Cert-Subject-Alt-Name-Dns    1931    string
ATTRIBUTE       TLS-Client-Cert-Subject-Alt-Name-Upn    1932    string
ATTRIBUTE       TLS-PSK-Identity                        1933    string
ATTRIBUTE       TLS-Client-Cert-Subject-Alt-Name-URI    1934    string

# 1934 - 1939: reserved for future cert attributes

# 1940 - 1949: reserved for TLS session caching, mostly in 3.1

# Set by EAP-TLS code
ATTRIBUTE       TLS-OCSP-Cert-Valid                     1943    integer
VALUE   TLS-OCSP-Cert-Valid             unknown                 3
VALUE   TLS-OCSP-Cert-Valid             skipped                 2
VALUE   TLS-OCSP-Cert-Valid             yes                     1
VALUE   TLS-OCSP-Cert-Valid             no                      0

                                                                                534,
```

   In freeradius the URI is retrieved from certificate using following attribute
   TLS-Client-Cert-Subject-Alt-Name-Uri

TLS.patch -- tls.patch has following modifications shown in below picture.



```
102a103,105
> /* Iot code change  */
> static char urI[1024];
>
1706c1709,1710
< static char const *cert_attr_names[8][2] = {
---
> /*iot code change*/
> static char const *cert_attr_names[9][2] = {
1714c1718,1721
<       { "TLS-Client-Cert-Subject-Alt-Name-Upn",    "TLS-Cert-Subject-Alt-Name-Upn" }
---
>       { "TLS-Client-Cert-Subject-Alt-Name-Upn",    "TLS-Cert-Subject-Alt-Name-Upn" },
>         /* Iot Code change */
>         { "TLS-Client-Cert-Subject-Alt-Name-URI",       "TLS-Cert-Subject-Alt-Name-URI"
}
>
1725c1732,1733
<
---
> /* Iot Code change */
> #define FR_TLS_SAN_URI          (8)
1759a1768,1769
>              /* Iot Code change */
>     static char           Uri[1024];
1906a1917,1926
>                                        /* Iot Code change */
> #ifdef GEN_URI
>                     case GEN_URI:
>                     vp = fr_pair_make(talloc_ctx, certs,
cert_attr_names[FR_TLS_SAN_URI][lookup],(char *) ASN1_STRING_data(name-
>d.uniformResourceIdentifier), T_OP_SET);
>                         Uri[sizeof(subject) - 1] = '\0';
>                         strcpy(Uri,( (char *) ASN1_STRING_data(name-
>d.uniformResourceIdentifier)));
>                         strcpy(urI,Uri);
>                         break;
> #endif  //GEN_URI
>
```

4. Run following commands to build freeradius source code
   $ *./configure*
   $ *make*
   $ *sudo make install*

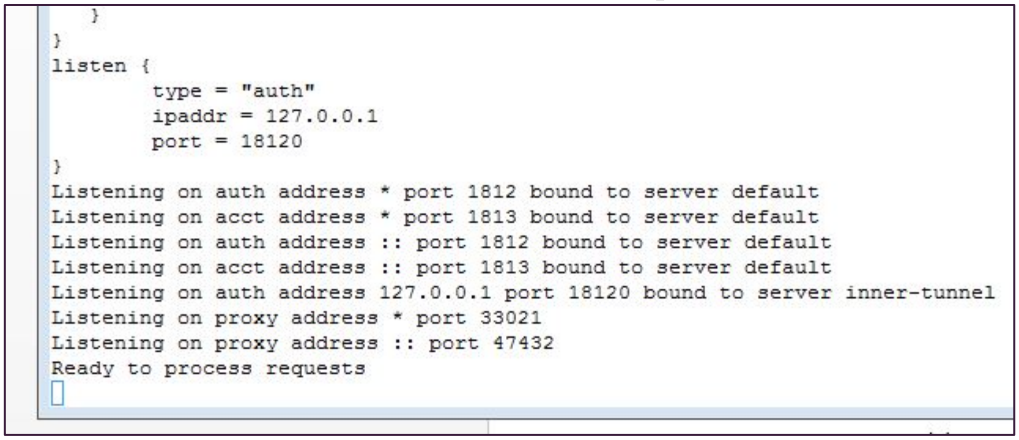
5. After finishing these steps FreeRADIUS should apply the changes that are made to tls.c

6. Starting-Up FreeRADIUS
   To run FreeRADIUS in debug mode, execute radiusd –X

Start FreeRADIUS server in debug mode

```
$ sudo radius –X
```

Freeradius should run and wait for the request as shown below



```
   }
}
listen {
        type = "auth"
        ipaddr = 127.0.0.1
        port = 18120
}
Listening on auth address * port 1812 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on auth address :: port 1812 bound to server default
Listening on acct address :: port 1813 bound to server default
Listening on auth address 127.0.0.1 port 18120 bound to server inner-tunnel
Listening on proxy address * port 33021
Listening on proxy address :: port 47432
Ready to process requests
```

To stop the server simply type **Ctrl + C**. This will stop the program.

NOTE: If you get an error message about port 1812 already in use, FreeRADIUS is already running kill the process and run again

```
$ pid of radius
$ kill -9 $(process id)
```

Or you can use **killall radius** to stop all services

To test the MUD CONTROLLER

Below commands will test if **mud_controller.py** script downloading MUD file from mudserver

```
$ /usr/local/etc/controller$ python mud_controller.py [MUD_URI/NULL] [R/W/U]
```

W → Get the mud file and signature file from mud uri and verify the signature and store the MUD file.

```
$ /usr/local/etc/controller$ python mud_controller.py http://<muduri>/mud/device1.json W
```

R1→ For INGRESS

```
$ /usr/local/etc/controller$ python mud_controller.py null R1
```

R2 → For EGRESS

```
$ /usr/local/etc/controller$ python mud_controller.py null R2
```

U1 → For INGRESS ACE Name

```
$ /usr/local/etc/controller$ python mud_controller.py null U1
```

U2 → For EGRESS ACE Name

```
$ /usr/local/etc/controller$ python mud_controller.py null U2
```

Testing using RADCLIENT at Client Side

Run following command

- ⇒ **With MUD URI attribute**

```
$ echo "User-Name=<username>, User-Password=<password>, Cisco-MUD-URI=http://<ipaddress of mudserver>/mud/Device1.json " | /usr/local/bin/radclient -x localhost:1812 auth testing123
```

Section output:

RADCLIENT TEST USING PAP Authentication

MUD URI: *http://eapm-lnx-05.cisco.com/.well-known/mud/Device1.json*
User Name: bob
Password: testing
MUD file name: Device1.json

```
$ echo "User-Name=bob, User-Password=testing, Cisco-MUD-URI=http://eapm-lnx-05.cisco.com/.well-known/mud/Device1.json " | /usr/local/bin/radclient -x localhost:1812 auth testing123
```

Output:

```
Sent Access-Request Id 99 from 0.0.0.0:42556 to 127.0.0.1:1812 length 107
        User-Name = "bob"
        User-Password = "testing"
        Cisco-MUD-URI = "http://<mud_ipaddress>:8000/mud/Device1.json"
        Cleartext-Password = "testing"
Received Access-Accept Id 99 from 127.0.0.1:1812 to 0.0.0.0:0 length 122
        Cisco-AVPair = "ACS:CiscoSecure-Defined-ACL=led_control.in"
        Cisco-AVPair = "ACS:CiscoSecure-Defined-ACL=temp_control.out"
```

Freeradius Output:

```
(0) Received Access-Request Id 99 from 127.0.0.1:42556 to 127.0.0.1:1812 length 107
(0)   User-Name = "bob"
(0)   User-Password = "testing"
(0)   Cisco-MUD-URI = "http://<mud_ipaddress>:8000/mud/Device1.json"
        .

(0) Sent Access-Accept Id 99 from 127.0.0.1:1812 to 127.0.0.1:42556 length 0
(0)   Cisco-AVPair := "ACS:CiscoSecure-Defined-ACL=led_control.in"
(0)   Cisco-AVPair += "ACS:CiscoSecure-Defined-ACL=temp_control.out"
(0) Finished request
```