AAA (ONOS 1.10)

- Introduction
- Configure ONOS
  - Activate AAA app
  - Provide Network Configuration
- Configure Radius Server
  - Install FreeRadius
  - Configure FreeRadius
    - Add a user
    - Allow external clients
    - Use TLS
  - Generate SSL Certificates
    - Generate CA certificate (ca.pem) and private key (privkey.pem)
    - Generate and sign server certificate (server.pem) and private key (server.key)
    - Generate and sign client certificate (client.pem) and private key (client.key)
    - Deploy keys and certificates
- Testing
  - Compose wpa_supplicant.conf
  - Run the test client
- Reference

Introduction

In this page, we will explain how to use Trellis Underlay Fabric with AAA service, which can be used to authenticate a client host. We will explain how this works with a simple single switch topology.

![Single Switch Topology](image)

Configure ONOS

Activate AAA app

We need to install and activate AAA app separately since it is located in a separate (CORD) repository. There are multiple methods to install and activate a pre-compiled app. Let's use CLI now.

<table>
<thead>
<tr>
<th>Install pre-built app</th>
</tr>
</thead>
<tbody>
<tr>
<td>onos-app localhost install! aaa-1.1-SNAPSHOT.oar</td>
</tr>
</tbody>
</table>
Note that this configuration can be used together with DHCP Relay Service. In that case we no longer need to configure host. Let's assume we don't have DHCP relay service now.

Provide Network Configuration

We need to provide AAA configuration in the `apps` section (for example, in a file named `aaa-conf.json`). We only need to specify the IP address of the Radius server and the secret.

The IP address of the Radius server
The UDP port of the Radius server. (Optional -- ONOS will use port 1812 by default).
The Radius secret. This needs to be consistent with the Radius server configuration

```json
Network Configuration

{
    "apps": {
        "org.opencord.aaa": {
            "AAA": {
                "radiusIp": "10.128.0.231",
                "radiusServerPort": "1812",
                "radiusSecret": "howdoyouturnthison"
            }
        }
    }
}
```

Upload the configuration to ONOS:

```
Upload AAA configuration

onos-netcfg $OC1 aaa-conf.json
```

Configure Radius Server

Install FreeRadius

Technically all Radius server should work. However, the way to configure them are probably different case to case. Here we use FreeRadius on Ubuntu as an example.

To install the Radius server, simply run:

```bash
Install FreeRadius Server

sudo apt-get install freeradius
```

Configure FreeRadius

Add a user

We usually connect Radius server to a database where we store the user information. In this article, we statically configure a user to simplify the setup.

To add a user `admin` with password `cord_test`, edit `/etc/freeradius/users` and add following lines:

```
Add a user

/etc/freeradius/users

admin  Cleartext-Password := "cord_test"
        Reply-Message = "Hello, %User-Name"
```
Allow external clients

By default the Radius server only accepts requests from localhost. To allow external clients, we need to modify `/etc/freeradius/clients.conf`

We also need to change the secret.

```
/etc/freeradius/clients.conf

-client localhost |
+client 0.0.0.0/0 |
-       secret          = testing123
+       secret          = howdoyouturnthison
```

Use TLS

By default, FreeRadius use MD5 challenge response to authenticate clients. To use TLS, we need to modify `/etc/freeradius/eap.conf`

We also need to change the private key password.

```
/etc/freeradius/eap.conf

-               default_eap_type = md5
+               default_eap_type = tls
-                       private_key_password = whatever
+                       private_key_password = onos_test
```

The key and certificates required by TLS will locate under `/etc/freeradius/certs` by default. There will be three symbolic links link to `ca.pem`, `server.key`, `server.pem`. We only need to change the symbolic links after we generates the keys and certificates. Therefore, we don't need to change the path in `/etc/freeradius/eap.conf`

Generate SSL Certificates

Both server certificate and client certificate need to be signed by the same CA certificate. Also note that each key we generate below needs a unique Common Name.

Generate CA certificate (ca.pem) and private key (privkey.pem)

```
Generate CA

openssl req -out ca.pem -new -x509
```

Generate and sign server certificate (server.pem) and private key (server.key)

```
Generate server certificate and key

openssl genrsa -out server.key 1024
openssl req -key server.key -new -out server.req
openssl x509 -req -in server.req -CA ca.pem -CAkey privkey.pem -CAserial file.srl -out server.pem
```

Generate and sign client certificate (client.pem) and private key (client.key)
Generate client certificate and key

```
openssl genrsa -out client.key 1024
openssl req -key client.key -new -out client.req
openssl x509 -req -in client.req -CA ca.pem -CAkey privkey.pem -CAserial file.srl -out client.pem
```

Deploy keys and certificates

On the server side, please link `/etc/freeradius/ca.pem, server.key, server.pem` to the files we just generated.
Also copy ca.pem, client.key, client.pem to the client side through a secured channel. They will later be used when testing the Radius authentication.

Testing

We can use the `wpa_supplicant` as the test client. In case `wpa_supplicant` has not been installed, you can run `sudo apt-get install wpasupplicant`

Compose `wpa_supplicant.conf`

```
wpa_supplicant.conf

ctrl_interface=/var/run/wpa_supplicant
eapol_version=1
ap_scan=0
fast_reauth=0
network={
    key_mgmt=WPA-EAP
eap=TLS
    identity="admin"
    password="cord_test"
    ca_cert="ca.pem"
    client_cert="client.pem"
    private_key="client.key"
    private_key_passwd="onos_test"
    eapol_flags=3
}
```

Run the test client

```
sudo wpa_supplicant -Dwired -ieth1 -cwpa_supplicant.conf
```

You should see the following message if authentication succeed:

```
Authentication Messages

Successfully initialized wpa_supplicant
eth1: Associated with 01:80:c2:00:00:03
eth1: CTRL-EVENT-EAP-STARTED EAP authentication started
eth1: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=13
eth1: CTRL-EVENT-EAP-METHOD EAP vendor 0 method 13 (TLS) selected
eth1: CTRL-EVENT-EAP-PEER-CERT depth=1 subject='/C=US/ST=CA/L=Menlo Park/O=ON.Lab/CN=ca.cord.lab/emailAddress=xxx@xxx.xxx'
eth1: CTRL-EVENT-EAP-PEER-CERT depth=0 subject='/C=US/ST=CA/L=Menlo Park/O=ON.Lab/CN=server.cord.lab/emailAddress=xxx@xxx.xxx'
eth1: CTRL-EVENT-EAP-SUCCESS EAP authentication completed successfully
```
If you are using a Linux VM to send out this authentication message, make sure the Linux kernel of your host machine is v3.2 or more recent, or your EAPOL messages will be eaten up.

Reference

https://www.vocal.com/secure-communication/eapol-extensible-authentication-protocol-over-lan/
http://dst.lbl.gov/~boverhof/openssl_certs.html