EAP AND CERTIFICATE BASED AUTHENTICATIONS

EAP and Certificate based authentications

WLAN Radius EAP EAP-TLS EAP-TTLS EAP-GTC 802.1X API OpenOTP Server OCSP NAC Network Access Control Wifi

1. Overview

This guide explains how to deploy certificate-based authentications for users and computers using 802.1x with RCDevs solutions. This solution can be applied to Wireless LAN / Wired LAN networks, RCDevs Web applications and also custom integrations like certificate-based authentication on your own website through OpenOTP APIs. It also describes how to implement EAP-TTLS authentication and certificate based authentication through OpenOTP APIs.

All integrations require at least WebADM product installed and running. Please, refer to WebADM Installation Guide and WebADM Manual

Important note

For all certificate-based authentications, an OCSP request is involved to check the certificate validity or if the presented certificate has not been revoked. This is not a configurable option. If you want to use and validate certificates issued by another CA than WebADM/Rsignd through RCDevs solutions, the certificate must contain the OCSP URL.

For certificates issued by WebADM/Rsignd, the OCSP check will be performed locally with an internal WebADM OCSP endpoint. Once the OCSP check is done, OpenOTP is involved to eventually apply the configured client policies. For certificates issued by another PKI than WebADM/Rsignd, Radius Bridge will try to found and read the OCSP URL value contained in the presented certificate. Radius Bridge will manage the certificate validation and trust with the CA configured to be used with Radius Bridge and the OCSP validation with the OCSP backend configured in the Users/Computers certificates. OpenOTP can not be used in that scenario after the EAP-TLS validation. That means you can not involve WebADM client policies for certificate-based authentication not issued by WebADM/Rsignd.

1.1 Supported Scenarios for Extended Authentication Protocols

For EAP protocols, the integration requires the Radius Bridge component provided by RCDevs. EAP-TLS, EAP-TTLS, EAP-GTC protocols are supported.

- **EAP-TLS with WebADM internal PKI** : Certificate issued by WebADM/Rsignd internal PKI. Certificate-based authentication based on CA trust + OCSP check + WebADM client policies applicable.
- **EAP-TLS with external PKI** : Certificate issued by an external PKI. Certificate-based authentication based on CA trust + OCSP check with external OCSP. The OCSP endpoint URL must be available in the certificate.
- **EAP-TTLS** : Username, passwords authentication. In that scenario, you can use LDAP username/password. WebADM client policies are applicable. Additionally, you can configure a policy to require an OTP (OTP concatenated to the LDAP password / Challenge mode not supported or through Push authentication).
- **EAP-GTC** : Username, passwords authentication. In that scenario, you can use LDAP username/password. WebADM client policies are applicable. Additionally, you can configure a policy to require an OTP (OTP concatenated to the LDAP password / Challenge mode not supported or through Push authentication).

2. Certificate-based authentications for RCDevs Web Applications and WebADM Admin portal

2.1 Issue your certificate from RCDevs Self-Services

All the required certificates can be downloaded either from the Self-Service portal or from the WebADM administrative interface. This guide explains the process using the Self-Service portal. First log into the Self-Service portal with your username, then select the PKI page.

On the PKI page, first, click the “Get WebADM CA Certificate” to CA Certificate.

Next, click the “Add new Certificate” button to generate a user certificate followed by clicking the “Download” button.

> ⚠ Note
>
> Please note that the certificate password is required in the following steps and it is only available on this page. It cannot be recovered later.

You should now have the required certificates for client configuration.

2.2 Issue your certificate from WebADM Administrator Portal

Login on the WebADM admin portal with your `super_admin` or `other_admin` account and click on the account you want to issue a new certificate in the left LDAP tree.

In the LDAP action box on the top, click on `Create Certificate` button and you are prompted to the following page:

Provide the validity you want for the certificate, the format of the export you prefer. To login on the WebADM Admin portal, the `certificate usage` must be set to `Admin`. Click `Create Cert` button and a certificate for that user is now created. You can click the Download button to download the certificate. Import the certificate in your Certificate store or your Web browser keychain (Mozilla Firefox). Once the certificate is imported, you can continue to the next steps to enable certificate-based

authentication on the WebADM admin portal.

2.3 Enable Certificate-based authentication on WebADM Administrator Portal

To enable the user certificate login on the WebADM Admin portal, you need to edit `webadm.conf` file with the following setting :

```
# Administrator Portal's authentication method.
# - PKI: Requires client certificate and login password.
# - UID: Requires domain name, login name and password.
# - DN: Requires login DN and password.
# - OTP: Like UID with an OTP challenge.
# - U2F: Like UID with a FIDO-U2F challenge.
# - MFA: Like UID with both OTP and FIDO-U2F challenge.
# Using certificates is the most secure login method. To use certificate login,
# you must log in WebADM and create a login certificate for your administrators.
# The UID mode requires a WebADM domain to exist and have its User Search Base
# set to the subtree where are located the administrator users. When using UID
# and if there is no domain existing in WebADM, the login mode is automatically
# forced to DN. You will also need to log in with the full user DN and set up
# a WebADM domain to be able to use the UID login mode.admin_auth UID
admin_auth PKI
```

Restart WebADM services for changes to be applied.

You can now access to WebADM Admin portal and you should be prompted to select the certificate you want to use.

I selected the certificate previously created.

My identity is auto-completed after certificate validation. I need to provide my LDAP password and then I am connected to the WebADM Admin portal.

2.4 Enable Certificate-based authentication on RCDevs Web Applications (Self-Services)

For Web Applications provided by RCDevs, you can enable the login through users' certificates. For enabling that feature, you need to log in as super_admin on the WebADM Admin portal. Click the `Applications` tab and then go to the

`Self-Services` category.

On the Web application you want to enable the PKI login, click `CONFIGURE` button. Look for the following setting across your web application configuration:

Once the feature is enabled, a valid certificate must be provided to access the web applications.

To log in on the web applications, the user must have a valid certificate. The certificate type must be `User`. It also works with the `Admin` certificate type for super_admin which already has a certificate to access to WebADM Admin portal. Checkpoint 1.2.1 for more information on how to issue a user certificate from the WebADM Admin portal. Users can also generate `User` certificate types by themself through web applications.

3. Certificate-based authentication for custom integrations (API integrations)

OpenOTP provides SOAP API methods that can be integrated wherever you want to authenticate users through user certificate. For OpenOTP to be able to validate the user certificates, you need to respect the following prerequisites:

- The certificate must be stored on the user account in the userCertificate attribute,
- The account must be activated in WebADM,
- The certificate can be issued by WebADM or another PKI, as soon as the certificate is stored on the user account, it can be used to authenticate the user.

Below, the description of the 2 methods which can be used for this purpose.

```
<!-- PKI Authentication Methods -->

<message name="openotpPKILoginRequest">
    <part name="certificate" type="xsd:string"/>
    <part name="client" type="xsd:string"/>
    <part name="source" type="xsd:string"/>
    <part name="settings" type="xsd:string"/>
    <part name="options" type="xsd:string"/>
    <part name="virtual" type="xsd:string"/>
</message>

<message name="openotpPKILoginResponse">
    <part name="code" type="xsd:integer"/>
    <part name="error" type="xsd:string"/>
    <part name="message" type="xsd:string"/>
    <part name="username" type="xsd:string"/>
    <part name="domain" type="xsd:string"/>
    <part name="data" type="xsd:string"/>
</message>
```

In that documentation, I use the SOAPUI tool to test a certificate-based authentication. What is performed by SOAPUI must be implemented on the client system you want to enable certificate-based authentication. For e.g, if you want to enable certificate-based authentication on your intranet, you must implement SOAP calls on your intranet web server and configure the login page of your intranet website to ask the users for their certificate. When the user will access the intranet through his web browser, he will have to provide the certificate issued for this purpose. The certificate will be passed through the users' web browser to your website and your website must fill in the user certificate into the `certificate` parameter of the SOAP API `PKI Authentication` method. The certificate must be filled to the SOAP API in PEM format.

Below, the OpenOTP logs for that authentication.

```
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] New openotpPKILogin SOAP
request
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] > Certificate: Default\valery
(46)
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] > Client ID: PKITestPolicy
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] > Source IP: 8.8.8.8
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] > Settings: 8.8.8.8
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] Registered openotpPKILogin
request
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] Resolved LDAP user:
CN=valery,OU=SUPAdmins,DC=support,DC=rcdevs,DC=com (cached)
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] Resolved source location: US
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] Started transaction lock for
user
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] Found user fullname: valery
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] Found 10 user settings:
EnableLogin=Yes
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] Updated user data
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] Sent login success response
```

As you can see in the logs, the authentication is a success.

If I remove the certificate from the user account, then the authentication is immediately rejected.

```
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] New openotpPKILogin SOAP
request
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] > Certificate: Default\valery
(46)
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] > Client ID: PKITestPolicy
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] > Source IP: 8.8.8.8
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] > Settings: 8.8.8.8
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] Registered openotpPKILogin
request
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] Resolved LDAP user:
CN=valery,OU=SUPAdmins,DC=support,DC=rcdevs,DC=com (cached)
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] Resolved source location: US
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] Started transaction lock for
user
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] Found user fullname: valery
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] Found 10 user settings:
EnableLogin=Yes
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] User has no certificate
registered
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:037D471C] Sent failure response
```

4. Radius Bridge configuration for EAP-TLS

4.1 Configure your EAP equipment as a Radius client in Radius Bridge

On your Radius Bridge server, edit the `/opt/radiusd/conf/clients.conf` and add a RADIUS client (with IP address, port and RADIUS secret) for your equipement supporting EAP protocols.

Example:

```
client NAC_CLIENT {
    ipaddr          = <Radius_Client_IP>
    secret          = testing123
}
```

Restart RB service for changes be reflected.

4.2 Enabling EAP-TLS in Radius Bridge

If you wish to use EAP-TLS with user certificates for authentication (instead of username+password), you have to enable the following settings in `/opt/radiusd/conf/radiusd.conf`

```
# EAP-TLS Wifi support
# Enable cert_support if you want to use EAP-TLS authentication for Enterprise Wifi.
# With EAP-TLS, the user authentication uses a user certificate only (no username,
password or OTP).
# The user certificates are generated in WebADM or the self-services. You need to
enable the OCSP
# service URL below with EAP-TLS.
cert_support = yes

# OCSP URL for EAP-TLS
# The WebADM OCSP service is used by RadiusBridge to check the revocation status of a
user certificate.
# A certificate is considered valid if it's not expired and present on the LDAP user
object. In WebADM
# the OCSP cache is refreshed every 6 hours. You can force OCSP refresh by purging the
license cache.
ocsp_url = "https://localhost/ocsp/"
```

The OCSP setting here must always point to your WebADM OCSP URL (localhost or IP address). Restart RB service for changes to be reflected.

5. Configure RADIUS AAA Server on your EAP equipement (Access Point, WLAN Controller, LAN Switch…)

In that documentation, we use Access Point and WLAN controller as an example. The next step is to configure wireless to use WPA2 Enterprise security mode and to define the RADIUS server as the authentication provider for your WLAN. The specific configuration depends on the brand and model of your WLAN equipment, two examples are provided by the chapters below.

5.1 For WLAN Access Point

If you have a standalone WLAN access point or router, without a centralized controller, you must configure the RADIUS server to each access point/router.

The below image provides an example of Cisco Linksys wireless router configuration.

5.2 For Cisco WLAN Controller

In this case, we add a RADIUS AAA Server configuration to your Cisco WLAN controller:

1. Login to the WLC GUI.
2. Click Security and RADIUS > Authentication.
3. In the RADIUS Authentication servers page appears, click New to add a new RADIUS Authentication Server.

4. Enter the RADIUS server corresponding to the Radius Bridge configuration in chapter 2.

Next, configure the WLAN networks and settings:

1. Open the WLANs page from the controller web interface.
2. Choose an existing or create a new WLAN.
3. In the “Security” tab, open the “AAA Servers” subtab.
4. Select the RADIUS server you configured as the “Authentication server”.
5. Click Apply to save your configuration.

6. EAP-TLS authentication with certificate issued by internal WebADM/Rsignd PKI

6.1 Certificate Generation

Have a look on point #2 of that documentation to generate a `User` certificate. User certificate can be self-generated by end-users through RCDevs web applications (Self-Desk, SelReg), by the helpdesk team through Helpdesk application or through WebADM Admin portal.

6.2 Login on Wifi controller configured with EAP-TLS

Once my certificate has been issued and imported in my key store, I can use it for EAP-TLS authentication. Found below, the description of the certificate that I will use for that test authentication :

I click to connect on the Wifi I configured in EAP-TLS and prompted to select the mode and the identity I want to use for the login. I choose EAP-TLS (else EAP-TTLS is involved) and the Identity (Roland)

Then I click Connect and few seconds after, I am connected on the Wifi.

6.3 EAP-TLS logs on Radius Bridge

In order to debug EAP protocols, you can start Radius Bridge in debug mode with the following command :

```
/opt/radiusd/bin/radiusd debug
```

Below, all logs on regarding the EAP negociation previously performed in debug mode :

```
(0) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
141
(0)   User-Name = "Default\\roland"
(0)   NAS-IP-Address = 192.168.4.250
(0)   Called-Station-Id = "586d8fa0308d"
```

```
(0)   Called-Station-Id = "586d8fa0308d"
(0)   Calling-Station-Id = "f40f2423e0c7"
(0)   NAS-Identifier = "586d8fa0308d"
(0)   NAS-Port = 4
(0)   Framed-MTU = 1400
(0)   NAS-Port-Type = Wireless-802.11
(0)   EAP-Message = 0x020000130144656661756c745c726f6c616e64
(0)   Message-Authenticator = 0x8cb30e6eb5c035d76b6a48ea62d0eba7
(0) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(0)   authorize {
(0) eap: Peer sent EAP Response (code 2) ID 0 length 19
(0) eap: Continuing tunnel setup
(0)     [eap] = ok
(0)   } # authorize = ok
(0) Found Auth-Type = EAP
(0) # Executing group from file /opt/radiusd/lib/radiusd.ini
(0)   Auth-Type EAP {
(0) eap: Peer sent packet with method EAP Identity (1)
(0) eap: Calling submodule eap_ttls to process data
(0) eap_ttls: (TLS) Initiating new session
(0) eap: Sending EAP Request (code 1) ID 1 length 6
(0) eap: EAP session adding &reply:State = 0xd85aceebd85bdb18
(0)     [eap] = handled
(0)   } # Auth-Type EAP = handled
(0) Using Post-Auth-Type Challenge
(0) Post-Auth-Type sub-section not found.  Ignoring.
(0) session-state: Saving cached attributes
(0)   Framed-MTU = 994
(0) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(0)   EAP-Message = 0x010100061520
(0)   Message-Authenticator = 0x00000000000000000000000000000000
(0)   State = 0xd85aceebd85bdb18ad985fadddd9ba17
(0) Finished request
Waking up in 9.9 seconds.
(0) Cleaning up request packet ID 0 with timestamp +742
(1) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
146
(1)   User-Name = "Default\\roland"
(1)   NAS-IP-Address = 192.168.4.250
(1)   Called-Station-Id = "586d8fa0308d"
(1)   Calling-Station-Id = "f40f2423e0c7"
(1)   NAS-Identifier = "586d8fa0308d"
(1)   NAS-Port = 4
(1)   Framed-MTU = 1400
(1)   State = 0xd85aceebd85bdb18ad985fadddd9ba17
(1)   NAS-Port-Type = Wireless-802.11
(1)   EAP-Message = 0x02010006030d
(1)   Message-Authenticator = 0x73cf20651a33a0e2a9bbb615311673c7
(1) Restoring &session-state
(1)   &session-state:Framed-MTU = 994
(1) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(1)   authorize {
(1) eap: Peer sent EAP Response (code 2) ID 1 length 6
```

```
(1) eap: Continuing tunnel setup
(1)     [eap] = ok
(1)   } # authorize = ok
(1) Found Auth-Type = EAP
(1) # Executing group from file /opt/radiusd/lib/radiusd.ini
(1)   Auth-Type EAP {
(1) eap: Expiring EAP session with state 0xd85aceebd85bdb18
(1) eap: Finished EAP session with state 0xd85aceebd85bdb18
(1) eap: Previous EAP request found for state 0xd85aceebd85bdb18, released from the
list
(1) eap: Peer sent packet with method EAP NAK (3)
(1) eap: Found mutually acceptable type TLS (13)
(1) eap: Calling submodule eap_tls to process data
(1) eap_tls: (TLS) Initiating new session
(1) eap_tls: (TLS) Setting verify mode to require certificate from client
(1) eap: Sending EAP Request (code 1) ID 2 length 6
(1) eap: EAP session adding &reply:State = 0xd85aceebd958c318
(1)     [eap] = handled
(1)   } # Auth-Type EAP = handled
(1) Using Post-Auth-Type Challenge
(1) Post-Auth-Type sub-section not found.  Ignoring.
(1) session-state: Saving cached attributes
(1)   Framed-MTU = 994
(1) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(1)   EAP-Message = 0x010200060d20
(1)   Message-Authenticator = 0x00000000000000000000000000000000
(1)   State = 0xd85aceebd958c318ad985fadddd9ba17
(1) Finished request
Waking up in 9.9 seconds.
(1) Cleaning up request packet ID 0 with timestamp +742
(2) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
301
(2)   User-Name = "Default\\roland"
(2)   NAS-IP-Address = 192.168.4.250
(2)   Called-Station-Id = "586d8fa0308d"
(2)   Calling-Station-Id = "f40f2423e0c7"
(2)   NAS-Identifier = "586d8fa0308d"
(2)   NAS-Port = 4
(2)   Framed-MTU = 1400
(2)   State = 0xd85aceebd958c318ad985fadddd9ba17
(2)   NAS-Port-Type = Wireless-802.11
(2)   EAP-Message =
0x020200a10d800000009716030100920100008e030360e5e1d43c1356e4e327db12aa19c43cbf67a94d1177b

(2)   Message-Authenticator = 0xde2398f23b590bdec148204f601493a4
(2) Restoring &session-state
(2)   &session-state:Framed-MTU = 994
(2) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(2)   authorize {
(2) eap: Peer sent EAP Response (code 2) ID 2 length 161
(2) eap: Continuing tunnel setup
(2)     [eap] = ok
```

```
(2)   } # authorize = ok
(2) Found Auth-Type = EAP
(2) # Executing group from file /opt/radiusd/lib/radiusd.ini
(2)   Auth-Type EAP {
(2) eap: Expiring EAP session with state 0xd85aceebd958c318
(2) eap: Finished EAP session with state 0xd85aceebd958c318
(2) eap: Previous EAP request found for state 0xd85aceebd958c318, released from the
list
(2) eap: Peer sent packet with method EAP TLS (13)
(2) eap: Calling submodule eap_tls to process data
(2) eap_tls: (TLS) EAP Peer says that the final record size will be 151 bytes
(2) eap_tls: (TLS) EAP Got all data (151 bytes)
(2) eap_tls: (TLS) Handshake state - before SSL initialization
(2) eap_tls: (TLS) Handshake state - Server before SSL initialization
(2) eap_tls: (TLS) Handshake state - Server before SSL initialization
(2) eap_tls: (TLS) recv TLS 1.3 Handshake, ClientHello
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read client hello
(2) eap_tls: (TLS) send TLS 1.2 Handshake, ServerHello
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write server hello
(2) eap_tls: (TLS) send TLS 1.2 Handshake, Certificate
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write certificate
(2) eap_tls: (TLS) send TLS 1.2 Handshake, ServerKeyExchange
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write key exchange
(2) eap_tls: (TLS) send TLS 1.2 Handshake, CertificateRequest
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write certificate request
(2) eap_tls: (TLS) send TLS 1.2 Handshake, ServerHelloDone
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write server done
(2) eap_tls: (TLS) Server : Need to read more data: SSLv3/TLS write server done
(2) eap_tls: (TLS) In Handshake Phase
(2) eap: Sending EAP Request (code 1) ID 3 length 1004
(2) eap: EAP session adding &reply:State = 0xd85aceebda59c318
(2)     [eap] = handled
(2)   } # Auth-Type EAP = handled
(2) Using Post-Auth-Type Challenge
(2) Post-Auth-Type sub-section not found.  Ignoring.
(2) session-state: Saving cached attributes
(2)   Framed-MTU = 994
(2) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(2)   EAP-Message =
0x010303ec0dc0000008ee160303003d0200003903033b6d140fdc958c73e51eb3c0cfbaea24f5211392e6b8
(2)   Message-Authenticator = 0x00000000000000000000000000000000
(2)   State = 0xd85aceebda59c318ad985fadddd9ba17
(2) Finished request
Waking up in 9.9 seconds.
(2) Cleaning up request packet ID 0 with timestamp +742
(3) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
146
(3)   User-Name = "Default\\roland"
(3)   NAS-IP-Address = 192.168.4.250
(3)   Called-Station-Id = "586d8fa0308d"
(3)   Calling-Station-Id = "f40f2423e0c7"
```

```
(3)   NAS-Identifier = "586d8fa0308d"
(3)   NAS-Port = 4
(3)   Framed-MTU = 1400
(3)   State = 0xd85aceebda59c318ad985fadddd9ba17
(3)   NAS-Port-Type = Wireless-802.11
(3)   EAP-Message = 0x020300060d00
(3)   Message-Authenticator = 0x645a819abd55681138a46984c41a7c9f
(3) Restoring &session-state
(3)   &session-state:Framed-MTU = 994
(3) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(3)   authorize {
(3) eap: Peer sent EAP Response (code 2) ID 3 length 6
(3) eap: Continuing tunnel setup
(3)     [eap] = ok
(3)   } # authorize = ok
(3) Found Auth-Type = EAP
(3) # Executing group from file /opt/radiusd/lib/radiusd.ini
(3)   Auth-Type EAP {
(3) eap: Expiring EAP session with state 0xd85aceebda59c318
(3) eap: Finished EAP session with state 0xd85aceebda59c318
(3) eap: Previous EAP request found for state 0xd85aceebda59c318, released from the
list
(3) eap: Peer sent packet with method EAP TLS (13)
(3) eap: Calling submodule eap_tls to process data
(3) eap_tls: (TLS) Peer ACKed our handshake fragment
(3) eap: Sending EAP Request (code 1) ID 4 length 1004
(3) eap: EAP session adding &reply:State = 0xd85aceebdb5ec318
(3)     [eap] = handled
(3)   } # Auth-Type EAP = handled
(3) Using Post-Auth-Type Challenge
(3) Post-Auth-Type sub-section not found.  Ignoring.
(3) session-state: Saving cached attributes
(3)   Framed-MTU = 994
(3) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(3)   EAP-Message =
0x010403ec0dc0000008ee343236313330313439 5a180f32303731303431343133303134395a303431193017

(3)   Message-Authenticator = 0x00000000000000000000000000000000
(3)   State = 0xd85aceebdb5ec318ad985fadddd9ba17
(3) Finished request
Waking up in 9.9 seconds.
(3) Cleaning up request packet ID 0 with timestamp +742
(4) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
146
(4)   User-Name = "Default\\roland"
(4)   NAS-IP-Address = 192.168.4.250
(4)   Called-Station-Id = "586d8fa0308d"
(4)   Calling-Station-Id = "f40f2423e0c7"
(4)   NAS-Identifier = "586d8fa0308d"
(4)   NAS-Port = 4
(4)   Framed-MTU = 1400
(4)   State = 0xd85aceebdb5ec318ad985fadddd9ba17
```

```
(4)   NAS-Port-Type = Wireless-802.11
(4)   EAP-Message = 0x020400060d00
(4)   Message-Authenticator = 0x32dbe7446a81c8eb2cb17ef766684480
(4) Restoring &session-state
(4)   &session-state:Framed-MTU = 994
(4) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(4)   authorize {
(4) eap: Peer sent EAP Response (code 2) ID 4 length 6
(4) eap: Continuing tunnel setup
(4)     [eap] = ok
(4)   } # authorize = ok
(4) Found Auth-Type = EAP
(4) # Executing group from file /opt/radiusd/lib/radiusd.ini
(4)   Auth-Type EAP {
(4) eap: Expiring EAP session with state 0xd85aceebdb5ec318
(4) eap: Finished EAP session with state 0xd85aceebdb5ec318
(4) eap: Previous EAP request found for state 0xd85aceebdb5ec318, released from the
list
(4) eap: Peer sent packet with method EAP TLS (13)
(4) eap: Calling submodule eap_tls to process data
(4) eap_tls: (TLS) Peer ACKed our handshake fragment
(4) eap: Sending EAP Request (code 1) ID 5 length 308
(4) eap: EAP session adding &reply:State = 0xd85aceebdc5fc318
(4)     [eap] = handled
(4)   } # Auth-Type EAP = handled
(4) Using Post-Auth-Type Challenge
(4) Post-Auth-Type sub-section not found.  Ignoring.
(4) session-state: Saving cached attributes
(4)   Framed-MTU = 994
(4) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(4)   EAP-Message =
0x010501340d80000008eeee8e2601ae3f80448a3fcc7735c5a67670614d68290275e0b7762b5af9b6d3d2afc

(4)   Message-Authenticator = 0x00000000000000000000000000000000
(4)   State = 0xd85aceebdc5fc318ad985fadddd9ba17
(4) Finished request
Waking up in 9.9 seconds.
(4) Cleaning up request packet ID 0 with timestamp +742
(5) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
1426
(5)   User-Name = "Default\\roland"
(5)   NAS-IP-Address = 192.168.4.250
(5)   Called-Station-Id = "586d8fa0308d"
(5)   Calling-Station-Id = "f40f2423e0c7"
(5)   NAS-Identifier = "586d8fa0308d"
(5)   NAS-Port = 4
(5)   Framed-MTU = 1400
(5)   State = 0xd85aceebdc5fc318ad985fadddd9ba17
(5)   NAS-Port-Type = Wireless-802.11
(5)   EAP-Message =
0x020504fc0dc0000007f316030306630b00065f00065c00030a30820306308201eea003020102020125300d0

(5)   Message-Authenticator = 0x0222e976218ffa92a2a60af5a6422107
```

```
(5)   Message-Authenticator = 0x0333c976318771e93e3ab0ef5ab432197
(5) Restoring &session-state
(5)   &session-state:Framed-MTU = 994
(5) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(5)   authorize {
(5) eap: Peer sent EAP Response (code 2) ID 5 length 1276
(5) eap: Continuing tunnel setup
(5)     [eap] = ok
(5)   } # authorize = ok
(5) Found Auth-Type = EAP
(5) # Executing group from file /opt/radiusd/lib/radiusd.ini
(5)   Auth-Type EAP {
(5) eap: Expiring EAP session with state 0xd85aceebdc5fc318
(5) eap: Finished EAP session with state 0xd85aceebdc5fc318
(5) eap: Previous EAP request found for state 0xd85aceebdc5fc318, released from the
list
(5) eap: Peer sent packet with method EAP TLS (13)
(5) eap: Calling submodule eap_tls to process data
(5) eap_tls: (TLS) EAP Peer says that the final record size will be 2035 bytes
(5) eap_tls: (TLS) EAP Expecting 2 fragments
(5) eap_tls: (TLS) EAP Got first TLS fragment (1266 bytes).  Peer says more fragments
will follow
(5) eap_tls: (TLS) EAP ACKing fragment, the peer should send more data.
(5) eap: Sending EAP Request (code 1) ID 6 length 6
(5) eap: EAP session adding &reply:State = 0xd85aceebdd5cc318
(5)     [eap] = handled
(5)   } # Auth-Type EAP = handled
(5) Using Post-Auth-Type Challenge
(5) Post-Auth-Type sub-section not found.  Ignoring.
(5) session-state: Saving cached attributes
(5)   Framed-MTU = 994
(5) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(5)   EAP-Message = 0x010600060d00
(5)   Message-Authenticator = 0x00000000000000000000000000000000
(5)   State = 0xd85aceebdd5cc318ad985fadddd9ba17
(5) Finished request
Waking up in 9.9 seconds.
(5) Cleaning up request packet ID 0 with timestamp +749
(6) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
921
(6)   User-Name = "Default\\roland"
(6)   NAS-IP-Address = 192.168.4.250
(6)   Called-Station-Id = "586d8fa0308d"
(6)   Calling-Station-Id = "f40f2423e0c7"
(6)   NAS-Identifier = "586d8fa0308d"
(6)   NAS-Port = 4
(6)   Framed-MTU = 1400
(6)   State = 0xd85aceebdd5cc318ad985fadddd9ba17
(6)   NAS-Port-Type = Wireless-802.11
(6)   EAP-Message =
0x020603070d00b7e2f7a30701ef63fcc94b0203010001a350304e301d0603551d0e0416041428a7dc1346e13

(6)   Message-Authenticator = 0x3a15445e85a724b07397c16d6dcda6bd
```

```
(6)   Message-Authenticator = 0x5a15445e65a724b0739fc1ddddcda0bd
(6) Restoring &session-state
(6)   &session-state:Framed-MTU = 994
(6) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(6)   authorize {
(6) eap: Peer sent EAP Response (code 2) ID 6 length 775
(6) eap: Continuing tunnel setup
(6)     [eap] = ok
(6)   } # authorize = ok
(6) Found Auth-Type = EAP
(6) # Executing group from file /opt/radiusd/lib/radiusd.ini
(6)   Auth-Type EAP {
(6) eap: Expiring EAP session with state 0xd85aceebdd5cc318
(6) eap: Finished EAP session with state 0xd85aceebdd5cc318
(6) eap: Previous EAP request found for state 0xd85aceebdd5cc318, released from the
list
(6) eap: Peer sent packet with method EAP TLS (13)
(6) eap: Calling submodule eap_tls to process data
(6) eap_tls: (TLS) EAP Got final fragment (769 bytes)
(6) eap_tls: (TLS) EAP Done initial handshake
(6) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write server done
(6) eap_tls: (TLS) recv TLS 1.2 Handshake, Certificate
(6) eap_tls: (TLS) Creating attributes from server certificate
(6) eap_tls:   TLS-Cert-Serial := "0ad37ee93fdbfe67f1115f96850d4495c8da6def"
(6) eap_tls:   TLS-Cert-Expiration := "20710414130149Z"
(6) eap_tls:   TLS-Cert-Subject := "/CN=WebADM CA #20034/O=Support RCDevs"
(6) eap_tls:   TLS-Cert-Issuer := "/CN=WebADM CA #20034/O=Support RCDevs"
(6) eap_tls:   TLS-Cert-Common-Name := "WebADM CA #20034"
(6) eap_tls: (TLS) Creating attributes from client certificate
(6) eap_tls:   TLS-Client-Cert-Serial := "25"
(6) eap_tls:   TLS-Client-Cert-Expiration := "220707170935Z"
(6) eap_tls:   TLS-Client-Cert-Subject :=
"/CN=Default\roland/UID=roland/DC=Default/description=USER"
(6) eap_tls:   TLS-Client-Cert-Issuer := "/CN=WebADM CA #20034/O=Support RCDevs"
(6) eap_tls:   TLS-Client-Cert-Common-Name := "Default\roland"
(6) eap_tls: Starting OCSP Request
(6) eap_tls: ocsp:  Using responder URL "https://192.168.4.20:443/ocsp/?
nosig=1&host=192.168.4.250&client=586d8fa0308d&source="
 This Update: Jul  7 17:18:19 2021 GMT
(6) eap_tls: ocsp: Cert status: good
(6) eap_tls: ocsp: Certificate is valid
(6) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read client certificate
(6) eap_tls: (TLS) recv TLS 1.2 Handshake, ClientKeyExchange
(6) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read client key exchange
(6) eap_tls: (TLS) recv TLS 1.2 Handshake, CertificateVerify
(6) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read certificate verify
(6) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read change cipher spec
(6) eap_tls: (TLS) recv TLS 1.2 Handshake, Finished
(6) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read finished
(6) eap_tls: (TLS) send TLS 1.2 ChangeCipherSpec
(6) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write change cipher spec
(6) eap_tls: (TLS) send TLS 1.2 Handshake, Finished
(6) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write finished
```

```
(6) eap_tls: (TLS) Handshake state - SSL negotiation finished successfully
(6) eap_tls: (TLS) Connection Established
(6) eap: Sending EAP Request (code 1) ID 7 length 61
(6) eap: EAP session adding &reply:State = 0xd85aceebde5dc318
(6)     [eap] = handled
(6)   } # Auth-Type EAP = handled
(6) Using Post-Auth-Type Challenge
(6) Post-Auth-Type sub-section not found.  Ignoring.
(6) session-state: Saving cached attributes
(6)   Framed-MTU = 994
(6) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(6)   EAP-Message =
0x0107003d0d8000000033140303000101160303002 8f3c248d2faf970c644015b2a2588a7eea20b7925835a1

(6)   Message-Authenticator = 0x00000000000000000000000000000000
(6)   State = 0xd85aceebde5dc318ad985fadddd9ba17
(6) Finished request
Waking up in 9.9 seconds.
(6) Cleaning up request packet ID 0 with timestamp +749
(7) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
146
(7)   User-Name = "Default\\roland"
(7)   NAS-IP-Address = 192.168.4.250
(7)   Called-Station-Id = "586d8fa0308d"
(7)   Calling-Station-Id = "f40f2423e0c7"
(7)   NAS-Identifier = "586d8fa0308d"
(7)   NAS-Port = 4
(7)   Framed-MTU = 1400
(7)   State = 0xd85aceebde5dc318ad985fadddd9ba17
(7)   NAS-Port-Type = Wireless-802.11
(7)   EAP-Message = 0x020700060d00
(7)   Message-Authenticator = 0xa5d3c08d8cc6b9f3daf805483b0c33af
(7) Restoring &session-state
(7)   &session-state:Framed-MTU = 994
(7) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(7)   authorize {
(7) eap: Peer sent EAP Response (code 2) ID 7 length 6
(7) eap: Continuing tunnel setup
(7)     [eap] = ok
(7)   } # authorize = ok
(7) Found Auth-Type = EAP
(7) # Executing group from file /opt/radiusd/lib/radiusd.ini
(7)   Auth-Type EAP {
(7) eap: Expiring EAP session with state 0xd85aceebde5dc318
(7) eap: Finished EAP session with state 0xd85aceebde5dc318
(7) eap: Previous EAP request found for state 0xd85aceebde5dc318, released from the
list
(7) eap: Peer sent packet with method EAP TLS (13)
(7) eap: Calling submodule eap_tls to process data
(7) eap_tls: (TLS) Peer ACKed our handshake fragment.  handshake is finished
Detected WebADM user certificate (calling OpenOTP)
USER
```

```
OpenOTP authentication succeeded
(7) eap: Sending EAP Success (code 3) ID 7 length 4
(7) eap: Freeing handler
(7)     [eap] = ok
(7)   } # Auth-Type EAP = ok
(7) Login OK: [Default\roland] (from client any port 4 cli f40f2423e0c7)
(7) Sent Access-Accept Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(7)   Reply-Message := "Authentication success"
(7)   MS-MPPE-Recv-Key =
0xe0d02a4251196b64032224dd80309d68f240db9a2f038e3b70e878f447c16f19
(7)   MS-MPPE-Send-Key =
0x13c4f8ae48ca72317a09c42288f80f0ec0cb3f491a0731bb95cf3db9ceda467f
(7)   EAP-Message = 0x03070004
(7)   Message-Authenticator = 0x00000000000000000000000000000000
(7)   User-Name = "Default\\roland"
(7) Finished request
Waking up in 9.9 seconds.
```

You can see at the end of the debug logs the following which confirm the authentication has been done successfully.

```
(7) Login OK: [Default\roland] (from client any port 4 cli f40f2423e0c7)
(7) Sent Access-Accept Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(7)   Reply-Message := "Authentication success"
(7)   MS-MPPE-Recv-Key =
0xe0d02a4251196b64032224dd80309d68f240db9a2f038e3b70e878f447c16f19
(7)   MS-MPPE-Send-Key =
0x13c4f8ae48ca72317a09c42288f80f0ec0cb3f491a0731bb95cf3db9ceda467f
(7)   EAP-Message = 0x03070004
(7)   Message-Authenticator = 0x00000000000000000000000000000000
(7)   User-Name = "Default\\roland"
```

Radius Bridge detected that the certificate used has been issued by WebADM PKI, then it called the WebADM OCSP to check the certificate revocation and after that, a login is started with OpenOTP in order to apply the client policy. The login with OpenOTP do not ask for any credentials, it is just a passthrough. In these logs, you can also see the OpenOTP authentication validated successfully after the handshake validation:

```
Detected WebADM user certificate (calling OpenOTP)
USER
OpenOTP authentication succeeded
```

6.4 WebADM OCSP and OpenOTP logs

Once the EAP negociation is successfully termininated, the certificate is checked with the corresponding OCSP endpoint. For a certificate issued by WebADM/Rsignd, the OSCP endpoint is local Below is the OCSP check performed during my test authentication :

```
[2021-07-07 19:18:19] [192.168.4.20:41938] [OCSP:LWAG0HBY] New OCSP request for serial
37 (GET)
[2021-07-07 19:18:19] [192.168.4.20:41938] [OCSP:LWAG0HBY] > Client ID: 586d8fa0308d
[2021-07-07 19:18:19] [192.168.4.20:41938] [OCSP:LWAG0HBY] > Client IP: 192.168.4.250
[2021-07-07 19:18:19] [192.168.4.20:41938] [OCSP:LWAG0HBY] Returning OCSP response
'Good'
```

If the certificate used during the authentication was expired or revoked by an administrator, you should see the following logs in webadm.log for the OCSP check :

```
[2021-07-08 10:24:12] [192.168.4.20:39226] [OCSP:9YFKUQ3J] New OCSP request for serial
37 (GET)
[2021-07-08 10:24:12] [192.168.4.20:39226] [OCSP:9YFKUQ3J] > Client ID: 586d8fa0308d
[2021-07-08 10:24:12] [192.168.4.20:39226] [OCSP:9YFKUQ3J] > Client IP: 192.168.4.250
[2021-07-08 10:24:12] [192.168.4.20:39226] [OCSP:9YFKUQ3J] Returning OCSP response
'Revoked'
```

Below is the OpenOTP authentication performed. You can easily identify in that logs that a client policy `Support Wifi` is involved.

```
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] New openotpSimpleLogin
SOAP request
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] > Username: roland
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] > Domain: Default
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] > Client ID: 586d8fa0308d
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] > Settings:
LoginMode=LDAP
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] > Options: RADIUS,-LDAP,-
U2F
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Enforcing client policy:
Support Wifi (matched client ID)
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Registered
openotpSimpleLogin request
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Resolved LDAP user:
CN=roland,OU=SUPAdmins,DC=support,DC=rcdevs,DC=com
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Started transaction lock
for user
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Found user fullname:
roland
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Found 52 user settings:
LoginMode=OTP,OTPType=TOKEN,PushLogin=Yes,ChallengeMode=No,ChallengeTimeout=90,OTPLength=
1:HOTP-SHA1-6:QN06-
T1M,DeviceType=FIDO2,U2FPINMode=Discouraged,SMSType=Normal,SMSMode=Ondemand,MailMode=Onde

[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Found 1 request settings:
LoginMode=LDAP
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Requested login factors:
None (passthrough)
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Updated user data
[2021-07-07 19:18:19] [192.168.4.20:33396] [OpenOTP:H9Y7ZVYR] Sent login success
response
```

7. EAP-TLS authentication with a certificate issued by external PKI

7.1 Certificate Generation

Please, refer to your PKI solution documentation to issue certificates.

7.2 Radius Bridge extra configuration

For Radius Bridge to be able to validate a certificate (user or computer certificates) issued by another PKI than WebADM, you must configure your external CA certificate(s) in `/opt/radiusd/conf/ca.crt`. If the CA certificate which issued the users/computers certificates used for authentication is not configured, then the CA trust verification will not work on the Radius side and the request will be rejected. In that case, the OCSP check is not performed. By default, only the WebADM CA certificate is configured in `ca.crt`. Add all CA certificates you need at the tail of this file.

See in the below example, the first certificate is my WebADM CA certificate and the following one is the CA certificate of my Windows PKI.

```
-----BEGIN CERTIFICATE-----
MIIDSDCCAjCgAwIBAgIUCtN+6T/b/mfxEV+WhQ1Elcjabe8wDQYJKoZIhvcNAQEL
BQAwNDEZMBcGA1UEAwwQV2ViQURNIENBICMyMDAzNDEXMBUGA1UECgwOU3VwcG9y
dCBSQ0RldnMwIBcNMjEwNDI2MTMwMTQ5WhgPMjA3MTA0MTQxMzAxNDlaMDQxGTAX
BgNVBAMMEFdlYkFETSBDQSAjMjAwMzQxFzAVBgNVBAoMDlN1cHBvcnQgUkNEZXZz
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvHDOR38WxtHY3lqoFaZb
DAVqE2Dr5RXHHvdRywKHPTI5Dt6MICCa8MBjGTlxXgmDolVxam5eYZpG8KhKpE1Q
zm5PbRryMeihM2GlrBn1Nm1oTIQXKEQ93bfa9TDTEOeEatHX4BTbSG8G5UBJFn7g
2oBkQSLDRUBQSNlJVLWem6B1WkFd3Bzo5WDR6flgRCmU4ACP7XMSLRikyByPIVnV
JL/pDjQSwfKblX6145/pfehlFqS5uopODHVKdZglUX2EgZFQjHbLDwRt3BNmDyjn
rhvbk7NK0f0tAEM1b4zUvyk3QFuhyukgdAagHJQK5+D0jL7rdc234vejBwHvY/zJ
SwIDAQABo1AwTjAdBgNVHQ4EFgQUKKfcE0bhMsDMFCG9dyYRfv4jBRcwHwYDVR0j
BBgwFoAUKKfcE0bhMsDMFCG9dyYRfv4jBRcwDAYDVR0TBAUwAwEB/zANBgkqhkiG
9w0BAQsFAAOCAQEAfjMticIXIRwi/RMS/05zojYLk8MQOOdluMBlyoIGKHIWQsKs
itmypQIfb5IVYzNP1iB3ABMTjlOgncbpNNcd5JSPG3n7x43RiiuZ+58Y1mxbzkTm
Eo8G3rVBLwil3nAHepEA0PzpsIgRFPl+rta46rISexoMeXPBTiksyc2pZXyaHBZF
VTfud8I6gdUA3aXLpcgHlfaKSGZVTJ30JipNZDKn89AFWslKjJxZoj/03YoveBL9
XzkQaGqJFxiCVJyfBYUYoJz4bVl2j07Uj2gA75eymbH18l2Dc9ZUetOge+q9Guy2
LEiIyXbWdB3/DWNpOqG1PZKMllFfpzgslcp15A==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIDpTCCAo2gAwIBAgIQVz4fdQJ2QZhGnJ2/KomwAjANBgkqhkiG9w0BAQsFADBZ
MRMwEQYKCZImiZPyLGQBGRYDY29tMRYwFAYKCZImiZPyLGQBGRYGcmNkZXZzMRcw
FQYKCZImiZPyLGQBGRYHc3VwcG9ydDERMA8GA1UEAxMIU1VQQ0FBRDIwHhcNMjEw
NzA2MTYxNTU5WhcNNDEwNzA2MTYyNTU4WjBZMRMwEQYKCZImiZPyLGQBGRYDY29t
MRYwFAYKCZImiZPyLGQBGRYGcmNkZXZzMRcwFQYKCZImiZPyLGQBGRYHc3VwcG9y
dDERMA8GA1UEAxMIU1VQQ0FBRDIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEK
AoIBAQDCVs3A/ksabY9Ljz+MhZbbQR4qZB6m/ofHdmzB+Am7Tsv4x9ulbB597jDQ
qNaOoW6QVtUGHZYoYFXswiteyibFUIGTEjdPhDxU/SYvmGnmwZ0bEHIlS4/xJYW/
MvO3ykEul6Ugo8zAVc5ABQ1CjHi+ZtuaLy+A9eyxAeFL1emxvlbUn2mQvmmbUOtX
Xt3c15iY33fAqgE1+pHm6E7MdUrVpESX5ynaQCgeL9BTgNbfuS00rk62UPoCx0q1
C/K9Dql7f7PQRntk56NKdeIb600/xRuUcDx8XK1uyua1HFHZS2q7pjwlZFG8aRV2
ly1S8PxvalPi9RzXEO9AX+Mft/6lAgMBAAGjaTBnMBMGCSsGAQQBgjcUAgQGHgQA
QwBBMA4GA1UdDwEB/wQEAwIBhjAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQWBBTa
8I08lOjn2sDzuzREii1NThrF3zAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0B
AQsFAAOCAQEAODLcWmSutShVG3gPpX5dCxOKGsfBeGJirpCrupSWeDy2oAsyZUfl
vww/JYibjBD2RxOOikA8sAtjMmivzvLWaHkTZVpxgTQEUqoHqi0WAkJjTvAZljvt
Oqxlso8HZPqZXAuVHW4wOZaLKeZSoE2qJjAuYfHVI7ZILqUHeQ7i28nuZvQJtnDw
z292RfmbFE6iL9UAIe7i6KdVS6Apx9PJyt8vjN80hLEc2h94KKCK2S/q6cSTbllP
JHMSm8SEGSHOAfdr6cVtLqH9FCxNQmyoPI8ISyQa6/HqrdwENoeExQ2XI059vkgy
/RjlEukzs8c3B7cw9C3ViG7AVPwxr0bosw==
-----END CERTIFICATE-----
```

7.3 Wifi Login

Once my certificate has been issued and imported into my key store, I can use it for EAP-TLS authentications. Found below, the

description of the certificate that I will use for the authentication :

I click to connect on the Wifi I configured in EAP-TLS and am prompted to select the mode and the identity I want to use for the login. I choose EAP-TLS (else EAP-TTLS is involved) and the Identity (Roland).

Then I click Connect and a few seconds after, I am connected to the Wifi.

7.4 EAP-TLS logs on Radius Bridge

To debug EAP protocols, you can start Radius Bridge in debug mode with the following command :

```
/opt/radiusd/bin/radiusd debug
```

Below, all logs on regarding the EAP negotiation previously performed in debug mode :

```
(0) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
161
(0)   User-Name = "yoann@support.rcdevs.org"
(0)   NAS-IP-Address = 192.168.4.250
(0)   Called-Station-Id = "586d8fa0308d"
(0)   Calling-Station-Id = "f40f2423e0c7"
(0)   NAS-Identifier = "586d8fa0308d"
(0)   NAS-Port = 4
(0)   Framed-MTU = 1400
(0)   NAS-Port-Type = Wireless-802.11
(0)   EAP-Message = 0x0200001d01796f616e6e40737570706f72742e7263646576732e6f7267
(0)   Message-Authenticator = 0x151e965fe7c01e0f975d522d196f9708
(0) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(0)   authorize {
(0) eap: Peer sent EAP Response (code 2) ID 0 length 29
(0) eap: Continuing tunnel setup
(0)     [eap] = ok
(0)   } # authorize = ok
(0) Found Auth-Type = EAP
(0) # Executing group from file /opt/radiusd/lib/radiusd.ini
(0)   Auth-Type EAP {
(0) eap: Peer sent packet with method EAP Identity (1)
(0) eap: Calling submodule eap_ttls to process data
(0) eap_ttls: (TLS) Initiating new session
(0) eap: Sending EAP Request (code 1) ID 1 length 6
(0) eap: EAP session adding &reply:State = 0xa3c591bfa3c48447
(0)     [eap] = handled
(0)   } # Auth-Type EAP = handled
(0) Using Post-Auth-Type Challenge
```

```
(0) Using Post-Auth-Type Challenge
(0) Post-Auth-Type sub-section not found.  Ignoring.
(0) session-state: Saving cached attributes
(0)   Framed-MTU = 994
(0) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(0)   EAP-Message = 0x010100061520
(0)   Message-Authenticator = 0x00000000000000000000000000000000
(0)   State = 0xa3c591bfa3c484479baef00182daeb34
(0) Finished request
Waking up in 9.9 seconds.
(0) Cleaning up request packet ID 0 with timestamp +4
(1) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
156
(1)   User-Name = "yoann@support.rcdevs.org"
(1)   NAS-IP-Address = 192.168.4.250
(1)   Called-Station-Id = "586d8fa0308d"
(1)   Calling-Station-Id = "f40f2423e0c7"
(1)   NAS-Identifier = "586d8fa0308d"
(1)   NAS-Port = 4
(1)   Framed-MTU = 1400
(1)   State = 0xa3c591bfa3c484479baef00182daeb34
(1)   NAS-Port-Type = Wireless-802.11
(1)   EAP-Message = 0x02010006030d
(1)   Message-Authenticator = 0xc95882cd89c9b543d6e58ce00f9900ca
(1) Restoring &session-state
(1)   &session-state:Framed-MTU = 994
(1) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(1)   authorize {
(1) eap: Peer sent EAP Response (code 2) ID 1 length 6
(1) eap: Continuing tunnel setup
(1)     [eap] = ok
(1)   } # authorize = ok
(1) Found Auth-Type = EAP
(1) # Executing group from file /opt/radiusd/lib/radiusd.ini
(1)   Auth-Type EAP {
(1) eap: Expiring EAP session with state 0xa3c591bfa3c48447
(1) eap: Finished EAP session with state 0xa3c591bfa3c48447
(1) eap: Previous EAP request found for state 0xa3c591bfa3c48447, released from the
list
(1) eap: Peer sent packet with method EAP NAK (3)
(1) eap: Found mutually acceptable type TLS (13)
(1) eap: Calling submodule eap_tls to process data
(1) eap_tls: (TLS) Initiating new session
(1) eap_tls: (TLS) Setting verify mode to require certificate from client
(1) eap: Sending EAP Request (code 1) ID 2 length 6
(1) eap: EAP session adding &reply:State = 0xa3c591bfa2c79c47
(1)     [eap] = handled
(1)   } # Auth-Type EAP = handled
(1) Using Post-Auth-Type Challenge
(1) Post-Auth-Type sub-section not found.  Ignoring.
(1) session-state: Saving cached attributes
(1)   Framed-MTU = 994
(1) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
```

```
(1) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(1)   EAP-Message = 0x010200060d20
(1)   Message-Authenticator = 0x00000000000000000000000000000000
(1)   State = 0xa3c591bfa2c79c479baef00182daeb34
(1) Finished request
Waking up in 9.9 seconds.
(1) Cleaning up request packet ID 0 with timestamp +4
(2) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
311
(2)   User-Name = "yoann@support.rcdevs.org"
(2)   NAS-IP-Address = 192.168.4.250
(2)   Called-Station-Id = "586d8fa0308d"
(2)   Calling-Station-Id = "f40f2423e0c7"
(2)   NAS-Identifier = "586d8fa0308d"
(2)   NAS-Port = 4
(2)   Framed-MTU = 1400
(2)   State = 0xa3c591bfa2c79c479baef00182daeb34
(2)   NAS-Port-Type = Wireless-802.11
(2)   EAP-Message =
0x020200a10d800000009716030100920100008e030360eee3dde592a9950315cc693e3790de70c1a839887f9

(2)   Message-Authenticator = 0x8e24d74fcb56cb4bb9e22bb6d01eab79
(2) Restoring &session-state
(2)   &session-state:Framed-MTU = 994
(2) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(2)   authorize {
(2) eap: Peer sent EAP Response (code 2) ID 2 length 161
(2) eap: Continuing tunnel setup
(2)     [eap] = ok
(2)   } # authorize = ok
(2) Found Auth-Type = EAP
(2) # Executing group from file /opt/radiusd/lib/radiusd.ini
(2)   Auth-Type EAP {
(2) eap: Expiring EAP session with state 0xa3c591bfa2c79c47
(2) eap: Finished EAP session with state 0xa3c591bfa2c79c47
(2) eap: Previous EAP request found for state 0xa3c591bfa2c79c47, released from the
list
(2) eap: Peer sent packet with method EAP TLS (13)
(2) eap: Calling submodule eap_tls to process data
(2) eap_tls: (TLS) EAP Peer says that the final record size will be 151 bytes
(2) eap_tls: (TLS) EAP Got all data (151 bytes)
(2) eap_tls: (TLS) Handshake state - before SSL initialization
(2) eap_tls: (TLS) Handshake state - Server before SSL initialization
(2) eap_tls: (TLS) Handshake state - Server before SSL initialization
(2) eap_tls: (TLS) recv TLS 1.3 Handshake, ClientHello
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read client hello
(2) eap_tls: (TLS) send TLS 1.2 Handshake, ServerHello
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write server hello
(2) eap_tls: (TLS) send TLS 1.2 Handshake, Certificate
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write certificate
(2) eap_tls: (TLS) send TLS 1.2 Handshake, ServerKeyExchange
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write key exchange
(2) eap_tls: (TLS) send TLS 1.2 Handshake, CertificateRequest
```

```
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write certificate request
(2) eap_tls: (TLS) send TLS 1.2 Handshake, ServerHelloDone
(2) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write server done
(2) eap_tls: (TLS) Server : Need to read more data: SSLv3/TLS write server done
(2) eap_tls: (TLS) In Handshake Phase
(2) eap: Sending EAP Request (code 1) ID 3 length 1004
(2) eap: EAP session adding &reply:State = 0xa3c591bfa1c69c47
(2)     [eap] = handled
(2)   } # Auth-Type EAP = handled
(2) Using Post-Auth-Type Challenge
(2) Post-Auth-Type sub-section not found.  Ignoring.
(2) session-state: Saving cached attributes
(2)   Framed-MTU = 994
(2) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(2)   EAP-Message =
0x010303ec0dc0000008ee160303003d020000390303485f7206969cdeb359b5b8ef1223f46a180760161bad5

(2)   Message-Authenticator = 0x00000000000000000000000000000000
(2)   State = 0xa3c591bfa1c69c479baef00182daeb34
(2) Finished request
Waking up in 9.9 seconds.
(2) Cleaning up request packet ID 0 with timestamp +4
(3) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
156
(3)   User-Name = "yoann@support.rcdevs.org"
(3)   NAS-IP-Address = 192.168.4.250
(3)   Called-Station-Id = "586d8fa0308d"
(3)   Calling-Station-Id = "f40f2423e0c7"
(3)   NAS-Identifier = "586d8fa0308d"
(3)   NAS-Port = 4
(3)   Framed-MTU = 1400
(3)   State = 0xa3c591bfa1c69c479baef00182daeb34
(3)   NAS-Port-Type = Wireless-802.11
(3)   EAP-Message = 0x020300060d00
(3)   Message-Authenticator = 0x4368c85a736e9eba8dd582067d8725d6
(3) Restoring &session-state
(3)   &session-state:Framed-MTU = 994
(3) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(3)   authorize {
(3) eap: Peer sent EAP Response (code 2) ID 3 length 6
(3) eap: Continuing tunnel setup
(3)     [eap] = ok
(3)   } # authorize = ok
(3) Found Auth-Type = EAP
(3) # Executing group from file /opt/radiusd/lib/radiusd.ini
(3)   Auth-Type EAP {
(3) eap: Expiring EAP session with state 0xa3c591bfa1c69c47
(3) eap: Finished EAP session with state 0xa3c591bfa1c69c47
(3) eap: Previous EAP request found for state 0xa3c591bfa1c69c47, released from the
list
(3) eap: Peer sent packet with method EAP TLS (13)
(3) eap: Calling submodule eap_tls to process data
```

```
(3) eap_tls: (TLS) Peer ACKed our handshake fragment
(3) eap: Sending EAP Request (code 1) ID 4 length 1004
(3) eap: EAP session adding &reply:State = 0xa3c591bfa0c19c47
(3)     [eap] = handled
(3)   } # Auth-Type EAP = handled
(3) Using Post-Auth-Type Challenge
(3) Post-Auth-Type sub-section not found.  Ignoring.
(3) session-state: Saving cached attributes
(3)   Framed-MTU = 994
(3) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(3)   EAP-Message =
0x010403ec0dc0000008ee343236313303031343 95a180f32303731303431343133303134395a303431193017

(3)   Message-Authenticator = 0x00000000000000000000000000000000
(3)   State = 0xa3c591bfa0c19c479baef00182daeb34
(3) Finished request
Waking up in 9.9 seconds.
(3) Cleaning up request packet ID 0 with timestamp +4
(4) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
156
(4)   User-Name = "yoann@support.rcdevs.org"
(4)   NAS-IP-Address = 192.168.4.250
(4)   Called-Station-Id = "586d8fa0308d"
(4)   Calling-Station-Id = "f40f2423e0c7"
(4)   NAS-Identifier = "586d8fa0308d"
(4)   NAS-Port = 4
(4)   Framed-MTU = 1400
(4)   State = 0xa3c591bfa0c19c479baef00182daeb34
(4)   NAS-Port-Type = Wireless-802.11
(4)   EAP-Message = 0x020400060d00
(4)   Message-Authenticator = 0x9aceb3a87b07949021a95043f47537c4
(4) Restoring &session-state
(4)   &session-state:Framed-MTU = 994
(4) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(4)   authorize {
(4) eap: Peer sent EAP Response (code 2) ID 4 length 6
(4) eap: Continuing tunnel setup
(4)     [eap] = ok
(4)   } # authorize = ok
(4) Found Auth-Type = EAP
(4) # Executing group from file /opt/radiusd/lib/radiusd.ini
(4)   Auth-Type EAP {
(4) eap: Expiring EAP session with state 0xa3c591bfa0c19c47
(4) eap: Finished EAP session with state 0xa3c591bfa0c19c47
(4) eap: Previous EAP request found for state 0xa3c591bfa0c19c47, released from the
list
(4) eap: Peer sent packet with method EAP TLS (13)
(4) eap: Calling submodule eap_tls to process data
(4) eap_tls: (TLS) Peer ACKed our handshake fragment
(4) eap: Sending EAP Request (code 1) ID 5 length 308
(4) eap: EAP session adding &reply:State = 0xa3c591bfa7c09c47
(4)     [eap] = handled
```

```
(4)   } # Auth-Type EAP = handled
(4) Using Post-Auth-Type Challenge
(4) Post-Auth-Type sub-section not found.  Ignoring.
(4) session-state: Saving cached attributes
(4)   Framed-MTU = 994
(4) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(4)   EAP-Message =
0x010501340d80000008eeceb43790f97117393943a57b9c22f4b706e5c3961b0dc33f89818274bf63b49cf0c

(4)   Message-Authenticator = 0x00000000000000000000000000000000
(4)   State = 0xa3c591bfa7c09c479baef00182daeb34
(4) Finished request
Waking up in 9.9 seconds.
(4) Cleaning up request packet ID 0 with timestamp +4
(5) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
1436
(5)   User-Name = "yoann@support.rcdevs.org"
(5)   NAS-IP-Address = 192.168.4.250
(5)   Called-Station-Id = "586d8fa0308d"
(5)   Calling-Station-Id = "f40f2423e0c7"
(5)   NAS-Identifier = "586d8fa0308d"
(5)   NAS-Port = 4
(5)   Framed-MTU = 1400
(5)   State = 0xa3c591bfa7c09c479baef00182daeb34
(5)   NAS-Port-Type = Wireless-802.11
(5)   EAP-Message =
0x020504fc0dc000000bd31603030a430b000a3f000a3c00068d3082068930820571a00302010202131c00000

(5)   Message-Authenticator = 0xd35dd29be84dfdcc62ce92e26e0ee9a1
(5) Restoring &session-state
(5)   &session-state:Framed-MTU = 994
(5) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(5)   authorize {
(5) eap: Peer sent EAP Response (code 2) ID 5 length 1276
(5) eap: Continuing tunnel setup
(5)     [eap] = ok
(5)   } # authorize = ok
(5) Found Auth-Type = EAP
(5) # Executing group from file /opt/radiusd/lib/radiusd.ini
(5)   Auth-Type EAP {
(5) eap: Expiring EAP session with state 0xa3c591bfa7c09c47
(5) eap: Finished EAP session with state 0xa3c591bfa7c09c47
(5) eap: Previous EAP request found for state 0xa3c591bfa7c09c47, released from the
list
(5) eap: Peer sent packet with method EAP TLS (13)
(5) eap: Calling submodule eap_tls to process data
(5) eap_tls: (TLS) EAP Peer says that the final record size will be 3027 bytes
(5) eap_tls: (TLS) EAP Expecting 3 fragments
(5) eap_tls: (TLS) EAP Got first TLS fragment (1266 bytes).  Peer says more fragments
will follow
(5) eap_tls: (TLS) EAP ACKing fragment, the peer should send more data.
(5) eap: Sending EAP Request (code 1) ID 6 length 6
```

```
(5) eap: EAP session adding &reply:State = 0xa3c591bfa6c39c47
(5)     [eap] = handled
(5)   } # Auth-Type EAP = handled
(5) Using Post-Auth-Type Challenge
(5) Post-Auth-Type sub-section not found.  Ignoring.
(5) session-state: Saving cached attributes
(5)   Framed-MTU = 994
(5) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(5)   EAP-Message = 0x010600060d00
(5)   Message-Authenticator = 0x00000000000000000000000000000000
(5)   State = 0xa3c591bfa6c39c479baef00182daeb34
(5) Finished request
Waking up in 9.9 seconds.
(5) Cleaning up request packet ID 0 with timestamp +4
(6) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
1436
(6)   User-Name = "yoann@support.rcdevs.org"
(6)   NAS-IP-Address = 192.168.4.250
(6)   Called-Station-Id = "586d8fa0308d"
(6)   Calling-Station-Id = "f40f2423e0c7"
(6)   NAS-Identifier = "586d8fa0308d"
(6)   NAS-Port = 4
(6)   Framed-MTU = 1400
(6)   State = 0xa3c591bfa6c39c479baef00182daeb34
(6)   NAS-Port-Type = Wireless-802.11
(6)   EAP-Message =
0x020604fc0d40436c6173733d6365727469666696361746 96f6e417574686f72697479303106082b060105050

(6)   Message-Authenticator = 0xc36a157cb9203e93ec3f0559bf52f783
(6) Restoring &session-state
(6)   &session-state:Framed-MTU = 994
(6) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(6)   authorize {
(6) eap: Peer sent EAP Response (code 2) ID 6 length 1276
(6) eap: Continuing tunnel setup
(6)     [eap] = ok
(6)   } # authorize = ok
(6) Found Auth-Type = EAP
(6) # Executing group from file /opt/radiusd/lib/radiusd.ini
(6)   Auth-Type EAP {
(6) eap: Expiring EAP session with state 0xa3c591bfa6c39c47
(6) eap: Finished EAP session with state 0xa3c591bfa6c39c47
(6) eap: Previous EAP request found for state 0xa3c591bfa6c39c47, released from the
list
(6) eap: Peer sent packet with method EAP TLS (13)
(6) eap: Calling submodule eap_tls to process data
(6) eap_tls: (TLS) EAP Got additional fragment (1270 bytes).  Peer says more fragments
will follow
(6) eap_tls: (TLS) EAP ACKing fragment, the peer should send more data.
(6) eap: Sending EAP Request (code 1) ID 7 length 6
(6) eap: EAP session adding &reply:State = 0xa3c591bfa5c29c47
(6)     [eap] = handled
(6)   } # Auth-Type EAP = handled
```

```
(6)   } # Auth-Type EAP = handled
(6) Using Post-Auth-Type Challenge
(6) Post-Auth-Type sub-section not found.  Ignoring.
(6) session-state: Saving cached attributes
(6)   Framed-MTU = 994
(6) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(6)   EAP-Message = 0x010700060d00
(6)   Message-Authenticator = 0x00000000000000000000000000000000
(6)   State = 0xa3c591bfa5c29c479baef00182daeb34
(6) Finished request
Waking up in 9.9 seconds.
(6) Cleaning up request packet ID 0 with timestamp +4
(7) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
649
(7)   User-Name = "yoann@support.rcdevs.org"
(7)   NAS-IP-Address = 192.168.4.250
(7)   Called-Station-Id = "586d8fa0308d"
(7)   Calling-Station-Id = "f40f2423e0c7"
(7)   NAS-Identifier = "586d8fa0308d"
(7)   NAS-Port = 4
(7)   Framed-MTU = 1400
(7)   State = 0xa3c591bfa5c29c479baef00182daeb34
(7)   NAS-Port-Type = Wireless-802.11
(7)   EAP-Message =
0x020701f10d00df2f8cdf3484b11cda1f7828a08ad92feae9c4936e594f2473129bc4841921ce01f76be9c56

(7)   Message-Authenticator = 0x1002cbaebad594dc4ec007166974e7b8
(7) Restoring &session-state
(7)   &session-state:Framed-MTU = 994
(7) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(7)   authorize {
(7) eap: Peer sent EAP Response (code 2) ID 7 length 497
(7) eap: Continuing tunnel setup
(7)     [eap] = ok
(7)   } # authorize = ok
(7) Found Auth-Type = EAP
(7) # Executing group from file /opt/radiusd/lib/radiusd.ini
(7)   Auth-Type EAP {
(7) eap: Expiring EAP session with state 0xa3c591bfa5c29c47
(7) eap: Finished EAP session with state 0xa3c591bfa5c29c47
(7) eap: Previous EAP request found for state 0xa3c591bfa5c29c47, released from the
list
(7) eap: Peer sent packet with method EAP TLS (13)
(7) eap: Calling submodule eap_tls to process data
(7) eap_tls: (TLS) EAP Got final fragment (491 bytes)
(7) eap_tls: (TLS) EAP Done initial handshake
(7) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write server done
(7) eap_tls: (TLS) recv TLS 1.2 Handshake, Certificate
(7) eap_tls: (TLS) Creating attributes from server certificate
(7) eap_tls:   TLS-Cert-Serial := "573e1f7502764198469c9dbf2a89b002"
(7) eap_tls:   TLS-Cert-Expiration := "410706162558Z"
(7) eap_tls:   TLS-Cert-Subject := "/DC=com/DC=rcdevs/DC=support/CN=SUPCAAD2"
(7) eap_tls:   TLS-Cert-Issuer := "/DC=com/DC=rcdevs/DC=support/CN=SUPCAAD2"
```

```
(7) eap_tls:   TLS-Cert-Issuer := "/DC=com/DC=rcdevs/DC=support/CN=SUPCAAD2
(7) eap_tls:   TLS-Cert-Common-Name := "SUPCAAD2"
(7) eap_tls: (TLS) Creating attributes from client certificate
(7) eap_tls:   TLS-Client-Cert-Serial := "1c000000071f3a399047c83195000000000007"
(7) eap_tls:   TLS-Client-Cert-Expiration := "220707124830Z"
(7) eap_tls:   TLS-Client-Cert-Subject :=
"/DC=com/DC=rcdevs/DC=support/OU=SUPAdmins/CN=yoann
traut/emailAddress=yoann@suport.rcdevs.org"
(7) eap_tls:   TLS-Client-Cert-Issuer := "/DC=com/DC=rcdevs/DC=support/CN=SUPCAAD2"
(7) eap_tls:   TLS-Client-Cert-Common-Name := "yoann traut"
(7) eap_tls:   TLS-Client-Cert-Subject-Alt-Name-Upn := "yoann@support.rcdevs.org"
(7) eap_tls:   TLS-Client-Cert-Subject-Alt-Name-Email := "yoann@support.rcdevs.com"
(7) eap_tls:   TLS-Client-Cert-X509v3-Extended-Key-Usage += "Microsoft Trust List
Signing, Microsoft Encrypted File System, E-mail Protection, TLS Web Client
Authentication"
(7) eap_tls:   TLS-Client-Cert-X509v3-Subject-Key-Identifier +=
"08:B8:8A:89:0B:4F:FB:85:19:AC:FC:57:80:50:F9:3F:35:18:72:A4"
(7) eap_tls:   TLS-Client-Cert-X509v3-Authority-Key-Identifier +=
"keyid:DA:F0:8D:3C:94:E8:E7:DA:C0:F3:BB:34:44:8A:2D:4D:4E:1A:C5:DF\n"
(7) eap_tls: Starting OCSP Request
(7) eap_tls: WARNING: ocsp: Using client certificate OCSP service "AD19-
2.support.rcdevs.com:35502400"
(7) eap_tls: ocsp:  Using responder URL "http://AD19-2.support.rcdevs.com:80/ocsp"
 This Update: Jul 13 16:16:00 2021 GMT
 Next Update: Jul 15 04:36:00 2021 GMT
(7) eap_tls: ocsp: Cert status: good
(7) eap_tls: ocsp: Certificate is valid
(7) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read client certificate
(7) eap_tls: (TLS) recv TLS 1.2 Handshake, ClientKeyExchange
(7) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read client key exchange
(7) eap_tls: (TLS) recv TLS 1.2 Handshake, CertificateVerify
(7) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read certificate verify
(7) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read change cipher spec
(7) eap_tls: (TLS) recv TLS 1.2 Handshake, Finished
(7) eap_tls: (TLS) Handshake state - Server SSLv3/TLS read finished
(7) eap_tls: (TLS) send TLS 1.2 ChangeCipherSpec
(7) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write change cipher spec
(7) eap_tls: (TLS) send TLS 1.2 Handshake, Finished
(7) eap_tls: (TLS) Handshake state - Server SSLv3/TLS write finished
(7) eap_tls: (TLS) Handshake state - SSL negotiation finished successfully
(7) eap_tls: (TLS) Connection Established
(7) eap: Sending EAP Request (code 1) ID 8 length 61
(7) eap: EAP session adding &reply:State = 0xa3c591bfa4cd9c47
(7)     [eap] = handled
(7)   } # Auth-Type EAP = handled
(7) Using Post-Auth-Type Challenge
(7) Post-Auth-Type sub-section not found.  Ignoring.
(7) session-state: Saving cached attributes
(7)   Framed-MTU = 994
(7) Sent Access-Challenge Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(7)   EAP-Message =
0x0108003d0d800000003314030300010116030300288536cf955ae127cc305836d01e4d48ced9d478970e35a
```

```
(7)   Message-Authenticator = 0x00000000000000000000000000000000
(7)   State = 0xa3c591bfa4cd9c479baef00182daeb34
(7) Finished request
Waking up in 9.9 seconds.
(7) Cleaning up request packet ID 0 with timestamp +4
(8) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length
156
(8)   User-Name = "yoann@support.rcdevs.org"
(8)   NAS-IP-Address = 192.168.4.250
(8)   Called-Station-Id = "586d8fa0308d"
(8)   Calling-Station-Id = "f40f2423e0c7"
(8)   NAS-Identifier = "586d8fa0308d"
(8)   NAS-Port = 4
(8)   Framed-MTU = 1400
(8)   State = 0xa3c591bfa4cd9c479baef00182daeb34
(8)   NAS-Port-Type = Wireless-802.11
(8)   EAP-Message = 0x020800060d00
(8)   Message-Authenticator = 0xd2c3cab602de2e02d77e19be99876286
(8) Restoring &session-state
(8)   &session-state:Framed-MTU = 994
(8) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(8)   authorize {
(8) eap: Peer sent EAP Response (code 2) ID 8 length 6
(8) eap: Continuing tunnel setup
(8)     [eap] = ok
(8)   } # authorize = ok
(8) Found Auth-Type = EAP
(8) # Executing group from file /opt/radiusd/lib/radiusd.ini
(8)   Auth-Type EAP {
(8) eap: Expiring EAP session with state 0xa3c591bfa4cd9c47
(8) eap: Finished EAP session with state 0xa3c591bfa4cd9c47
(8) eap: Previous EAP request found for state 0xa3c591bfa4cd9c47, released from the
list
(8) eap: Peer sent packet with method EAP TLS (13)
(8) eap: Calling submodule eap_tls to process data
(8) eap_tls: (TLS) Peer ACKed our handshake fragment.  handshake is finished
Not an admin or user certificate (bypassing OpenOTP)
(8) eap: Sending EAP Success (code 3) ID 8 length 4
(8) eap: Freeing handler
(8)     [eap] = ok
(8)   } # Auth-Type EAP = ok
(8) Login OK: [yoann@support.rcdevs.org] (from client any port 4 cli f40f2423e0c7)
(8) Sent Access-Accept Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
(8)   MS-MPPE-Recv-Key =
0x36d0737a48f86e2bb10d394034beb9478a28c798de400f120e0a3123b690bbe6
(8)   MS-MPPE-Send-Key =
0xbb635e291c8cf549d6f537bba254bd47360df675135e9bcf1b94f5c6c385b3d0
(8)   EAP-Message = 0x03080004
(8)   Message-Authenticator = 0x00000000000000000000000000000000
(8)   User-Name = "yoann@support.rcdevs.org"
(8) Finished request
```

You can see the following in RB debug logs :

Radius Bridge detected that the certificate used has been issued by another PKI than WebADM/Rsignd, then it bypasses the OpenOTP call :

```
Not an admin or user certificate (bypassing OpenOTP)
```

Radiusd called the OCSP endpoint configured in the certificate and checked the validity/revocation with the OCSP :

```
(7) eap_tls: Starting OCSP Request
(7) eap_tls: WARNING: ocsp: Using client certificate OCSP service "AD19-
2.support.rcdevs.com:35502400"
(7) eap_tls: ocsp:  Using responder URL "http://AD19-2.support.rcdevs.com:80/ocsp"
 This Update: Jul 13 16:16:00 2021 GMT
 Next Update: Jul 15 04:36:00 2021 GMT
(7) eap_tls: ocsp: Cert status: good
(7) eap_tls: ocsp: Certificate is valid
```

OCSP endpoint returned a success.

At the end of debug logs, you can see that the authentication has been done successfully :

```
(8)   } # Auth-Type EAP = ok
(8) Login OK: [yoann@support.rcdevs.org] (from client any port 4 cli f40f2423e0c7)
(8) Sent Access-Accept Id 0 from 192.168.4.20:1812 to 192.168.4.250:32768 length 0
```

We are successfully connected to the Wifi using EAP-TLS.

8. Other examples of client configuration for user certificate-based authentication (EAP-TLS)

The correct authentication protocol and settings must be configured also into your network clients.

8.1 Generating and Downloading required Certificates

Please refer to point #2 of that documentation to issue a certificate with internal WebADM/Rsignd PKI. For other PKI solutions, please refer to your solution documentation.

8.2 Windows PCs

The exact configuration depends on the Windows version used.

8.2.1 Windows 10

Windows 10 has native support for the required authentication protocols, so it is possible to use certificates for authentication without additional software.

First, we must install the CA certificate of your WebADM to the Windows client. Open the CA certificate in Windows and click “Install Certificate”.

Click “Next” on the following page in Certificate Import Wizard.

On the next page, select the certificate store in which the certificate should be installed. You must install the certificate in the “Trusted Root Certification Authorities”. Click “Next” followed by “Finish” on the next page.

Next, we install the user certificate downloaded from Self-Service to the Windows client. Open the user certificate, select “Current User” in the wizard and click “Next” two times.

When the wizard asks for the password for the certificate, input the password you’ve received in the Self-Service desk when downloading the certificate.

You can let Windows select the certificate store for the user certificate automatically. Click “Next” followed by “Finish”.

Now we can connect to the wireless network, find the network in question from your network connections and click “Connect”.

When you are prompted for username and password, select “Connect using a certificate”.

Choose the user certificate you’ve installed previously, click “OK” followed by “Connect”.

8.3 macOS / iOS

In macOS, open the downloaded user certificate to install it into the keychain. Input the password you've received from the Self-Service Desk.

Once the user certificate is in your keychain, you can select the wireless network to connect to. In the following screen, select "Mode: EAP-TLS", use the installed user certificate as the identity and click "Join".

8.4 Android

Android has native support of the required protocols, although this might depend on the specific version of Android. First, transfer the downloaded certificate to your phone, and then configure the wireless network.

8.5 Linux

Most Linux clients also have native support and the connection can be configured graphically. Below is a screenshot of Ubuntu 18.04 Network Manager.

⚠ Note

Don't forget to authorize the communication on the 1812 UDP port (default RADIUS port for the authentication) from your ASA system to your Radius Bridge instance at the firewall level.

9. Client Configuration for username+password Authentication (EAP-TTLS)

The correct authentication protocol and settings must be configured also into your wireless clients.

⚠ Note

To secure an EAP-TTLS connection it is critical that you configure your Radius Bridge /opt/radiusd/conf/ca.crt certificate to all the clients to authenticate the Radius server you are connecting to. Without this, the client is vulnerable to man in the middle attack and will disclose username and password to malicious access points!

9.1 Windows PCs

The exact configuration depends on the Windows version used.

9.1.1 Windows 10

Windows 10 has native support for the required authentication protocols, so it is possible to connect to the network directly. However, Microsoft has some special [requirements] (https://support.microsoft.com/en-us/help/814394/certificate-requirements-when-you-use-eap-tls-or-peap-with-eap-tls) for the server certificates used in this case.

The current version of Radius Bridge uses certificates generated by WebADM which will comply with these requirements, but older installations must recreate the certificates with an additional configuration. Below is a sample of the required commands that can be used for creating a new server certificate on an older Radius Bridge server.

```
cd /opt/radiusd/conf
echo -e "[ xpserver_ext]\nextendedKeyUsage = 1.3.6.1.5.5.7.3.1\n" > xpextensions
openssl genrsa -out radiusd.key 2048
openssl req -sha256 -new -key radiusd.key -out radiusd.csr -subj
"/CN=HOSTNAME/O=ORGNAME"
openssl x509 -req -days 3650 -in radiusd.csr -signkey radiusd.key -out radiusd.crt -
extensions xpserver_ext -extfile xpextensions
```

9.1.2 Windows 7

Windows 7 does not support the required EAP-TTLS authentication natively, so you must either use a 3rd party wireless client or install a WPA supplicant plugin. In this guide, we have used the [GÉANTLink] (https://github.com/Amebis/GEANTLink) plugin.

1. Download the correct version of the GÉANTLink [binary] (https://github.com/Amebis/GEANTLink/releases) (GEANTLink32.msi or GEANTLink64.msi).
2. Install the plugin on the Windows client.
3. Configure the WLAN settings as in the below images.

4. You must transfer the /opt/radiusd/conf/radiusd.crt from the Radius Bridge server and select it with “Add CA from File”.

9.2 macOS / iOS

Users cannot directly configure the correct settings in macOS and iOS, instead, the configuration must be created using Apple Configurator 2. Please see the below picture for a sample configuration.

9.3 Android

Android has native support of the required protocols. The Below picture provides a sample of the settings.

9.4 Linux

Most Linux clients also have native support and the connection can be configured graphically. Below is a screenshot of Ubuntu 17.10 Network Manager.

10. Client policies for EAP-TLS, EAP-TTLS logins

10.1 EAP-TLS policy

For certificates issued by WebADM/Rsignd PKI, OpenOTP is involved in the login process to apply WebADM client policies. For EAP-TLS, it is useless to customize OpenOTP settings like the login factors (LDAP, OTP, LDAPOTP) or the token type because these settings are not applied in EAP-TLS authentication. Anyway, it can be useful to create WebADM client policies for your EAP systems. For example, you can allow which user domain(s) is/are able to access that system, which group(s) is/are allowed to log in on that system, configure excluded days where the logins are not allowed on that system… Below, an example of the client policy I created for my Wifi access.

In my `Support Wifi` policy, I allowed my `Wifi_Users` group to log in and disallow my `Domain Admins` group. I configured allowed hours and days to access that system. I configured the `NAS-Identifier` sent by my EAP system in the `Client Name Aliase`. This is needed to match the WebADM client policy with the client system.

For certificates issued by an external PKI, OpenOTP can not be involved in the login process to apply WebADM client policies.

10.2 EAP-TTLS/EAP-GTC policy

For EAP-TTLS and EAP-GTC client systems, you can create WebADM/OpenOTP client policies. In that scenario, OpenOTP is involved to validate credentials provided by the user during the authentication. In that scenario, you can configure OpenOTP settings. The first OpenOTP setting you need to configure for these clients systems is the de-activation of the challenge mode support because it is not supported by EAP clients. You can choose which factor you want to validate with OpenOTP (LDAP, OTP, LDAPOTP) for EAP-TTLS/GTC logins :

If you choose the LDAPOTP login mode, you must provide the LDAP password and the OTP in concatenated mode during the authentication. Use that kind of login mode will prevent you to save credentials for that system because the OTP will not be valid anymore for the next authentication.

The push login is supported on that mode. It means you just need to provide an LDAP username and password during the authentication and then you will receive a push login request to finish the login process. LDAP credentials can be saved for the next login, you will just have to approve the push request for the next logins.

Another interesting feature that can be used here is the implementation of `applications passwords`. When the `applications passwords` feature is enabled for a system, users can use a password randomly generated by WebADM to log in on a specific system. `Application passwords` are generated per client policy and are unique for each user. `Applications passwords` can be configured under OpenOTP configuration and can be generated by end-users through RCDevs self-services.

Application password and OpenOTP login mode

Applications passwords are not entering in conflict with the `Login Mode` configured in OpenOTP. For e.g, if the `Login Mode` is configured to `LDAPOTP`, users can log in using their `application password` or the LDAP and OTP passwords/Push.

E.g of `application password` generated for my `Wifi Support` client policy through the User Self-Service Desk.

Usage of my Application password to login :

I am successfully authenticated with my `application password`.

10.2.1 OpenOTP logs for login with an application password

Below, the WebADM/OpenOTP logs regarding the authentication performed with my application password.

```
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] New openotpSimpleLogin
SOAP request
```

```
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] > Username: yoann
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] > Password:
xxxxxxxxxxxxxxxxxxx
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] > Client ID: 586d8fa0308d
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] > Settings:
OpenOTP.ChallengeMode=No
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] > Options:
RADIUS,NOVOICE,-U2F
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Enforcing client policy:
Support Wifi (matched client ID)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Registered
openotpSimpleLogin request
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Checking OpenOTP license
for RCDevs Support
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] License Ok (34/50 active
users)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Resolved LDAP user:
CN=yoann traut,OU=SUPAdmins,DC=support,DC=rcdevs,DC=com (cached)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Resolved LDAP groups:
super_admin,domain admins,schema admins,administrators,denied rodc password replication
group
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Started transaction lock
for user
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Found user fullname:
yoann
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Found user language: FR
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Found 1 user emails:
support@rcdevs.com
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Found 52 user settings:
LoginMode=LDAPOTP,OTPType=TOKEN,PushLogin=Yes,ChallengeMode=No,ChallengeTimeout=90,OTPLer
1:HOTP-SHA1-6:QN06-
T1M,DeviceType=FIDO2,U2FPINMode=Discouraged,SMSType=Normal,SMSMode=Ondemand,MailMode=Onde
[2 Items]
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Found 1 request settings:
ChallengeMode=No
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Found 12 user data:
ListInit,ListState,AppKeyInit,Device1Type,Device1Name,Device1Data,Device1State,TokenType,

[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] OTP List present (0/25
passwords used)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Application passwords
present (valid until 2022-01-15 11:42:17)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Found 1 registered OTP
token (TOTP)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Challenge mode disabled
(assuming concatenated passwords)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Requested login factors:
AppKey | (LDAP & OTP)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Application password Ok
(Support Wifi)
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Returning 8 RADIUS reply
attributes
```

```
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Updated user data
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] Sent login success
response
```

11. Radius Return Attributes

Radius return attributes can be used with both EAP-TTLS and TLS starting from WebADM 1.7.9-1 and Radius Bridge 1.3.11. This is a powerful mechanism that allows you to centrally control various characteristics of the network connection on per user/group basis, for example:

-VLAN allocation

-Access Control List Configuration

-Quality of Service Policies

Please refer to your network equipment documentation on which attributes can be used for your specific use case. The related WebADM configuration is explained in the Radius Attributes guide.