EAP AUTHENTICATIONS

The specifications and information in this document are subject to change without notice. Companies, names, and data used in examples herein are fictitious unless otherwise noted. This document may not be copied or distributed by any means, in whole or in part, for any reason, without the express written permission of RCDevs Security.

WebADM and OpenOTP are trademarks of RCDevs. All further trademarks are the property of their respective owners.

No guarantee is given for the correctness of the information contained in this document. Please send any comments or corrections to info@rcdevs.com.

Limited Warranty - Copyright (c) 2010-2024 RCDevs Security SA. All Rights Reserved.

EAP Authentications

WLAN EAP EAP-TLS EAP-TTLS EAP-GTC 802.1X OCSP NAC Network Access Control Wifi Switch Router Port Based authentication WIFI Authentication LAN Authentication

1. Overview

This documentation provides comprehensive guidance on integrating RCDevs solutions with Extensible Authentication Protocol (EAP) methods for secure and efficient user and computer authentication. 802.1X is a specific IEEE standard that deals with network access control and authentication. It is used to ensure that only authorized devices and users can access a network. Here are the key points about 802.1X:

1. **Authentication**: 802.1X provides a framework for authenticating devices or users before they are granted access to a network. This authentication can involve various methods, including username and password, digital certificates, or other credentials.
2. **Port-Based Control**: The standard is often used in Ethernet networks, and it operates at the data link layer (Layer 2) of the OSI model. It enables the control of network access based on the physical port of the network switch to which a device is connected. This means that when a device is plugged into a port on a switch, it must authenticate itself before it's allowed to communicate on the network.
3. **Security**: 802.1X enhances network security by preventing unauthorized devices from connecting to the network. It helps protect against unauthorized access and potential security threats.
4. **EAP (Extensible Authentication Protocol)**: 802.1X typically uses EAP to handle the authentication process. EAP is a flexible authentication framework that supports various authentication methods.
5. In 802.1X authentication, there are three key components:
   - **Supplicant**: This is the device or user that's trying to gain access to the network. It initiates the authentication process.
   - **Authenticator**: This is the network device (e.g., a switch or wireless access point) that enforces authentication on the physical port.
   - **Authentication Server**: This is the server responsible for validating the credentials provided by the supplicant.
6. Role-Based Access Control: Once authentication is successful, 802.1X can also enforce role-based access control. This means that different users may have different levels of access or permissions within the network by assigning him a VLAN or a privilege level and more. Possibilities will depend on what it supported by your **Authenticator** equipment. That part is achieved by returning supported RADIUS attributes to the **authenticator** and where attribute(s) value(s) is/are configurable and stored centrally in WebADM users or groups. As an example, I can return a different VLAN assignation based on Users' groups membership.

Overall, 802.1X is widely used in enterprise networks and provides a robust mechanism for ensuring network security and controlling access to network resources.

RCDevs solutions offer a versatile platform for implementing robust authentication mechanisms that cater to a wide range of use cases. In this document, we will explore EAP authentication methods and configurations supported by RCDevs solutions but also the custom integrations of the OpenOTPPKILogin method which is also used by Radius Bridge to deals with OpenOTP and let OpenOTP validate the certificate.

Certificate based authentication involve a PKI service. The default and recommended setup which will support all features

described into that documentation is with the WebADM internal PKI service. That service is running in all WebADM infrastructure and is named Rsignd.

With that setup, you will be able to issue users and clients certificates which will be used for authentications.

Issued user certificates are always stored on the corresponding user object in the LDAP backend (in the userCertificate attribute).

Other type of certificates are stored in the SQL database configured with WebADM.

1.1 PKI Service

For a well-structured PKI infrastructure design, especially if you already have an operational PKI service within your infrastructure, we recommend configuring WebADM as a Subordinate Certificate Authority of your Enterprise CA as explained in that documentation. Additionally, please consult the WebADM administration guide for comprehensive information regarding the PKI service, including certificate issuance and management processes.

It is also possible to use a user certificate issued by an external PKI service, provided that the certificate is stored on the user's LDAP object in the 'userCertificate' attribute. However, there are some limitations associated with this scenario. One notable limitation is that EAP negotiation requires certificates issued by the same certificate authority (CA). If there is a mismatch between the CA of the certificates provided by the supplicant and the CA used by the authenticator, the EAP negotiation will fail. That scenario will work in custom integrations where you implement yourself the OpenOTPPKILogin API method and on RCDevs web portals (WebADM Admin GUI, WebApps logins).

1.2 Certificates

For computer authentications based on SSL certificates, the certificate must be a `Client` type when you are issuing it. For massive client certificate deployment, you can script your CSR and key generation and massively submit the CSRs to the Manager API of WebADM. The API method which can be used for this purpose is called *Sign_Certificate_Request*. The response will return the signed certificate in PEM format.

Sign certificate Request

Method: **Sign_Certificate_Request** | Returns: **String**

Required Parameters	Optional Parameters	
• **request** (String)	• **expires** (Integer)	Returns the locally signed certificate request (CSR) in PEM format.

Client certificates can be issued through WebADM Manager APIs or WebADM admin GUI only by a WebADM super_admin or an other_admin. Non-administrative users do not have the capability to issue client certificates, as this responsibility should remain within the purview of the IT/Security department(s).

For user authentication, the certificate must be a `User` type when you are issuing it. User certificate can be issued by the end-user itself through the Self-Services provided by RCDevs, from the WebADM Admin portal and from Manager API. For massive user certificate deployment, you can script your CSR and key generation and massively submit the CSRs to the Manager API of WebADM. The API method which can be used for this purpose is the same as for client certificate (Sign_Certificate_Request). The response will returned the signed certificate in PEM format.

Please, refer to the WebADM Administrator Guide to issue User, Server and client certificates.

1.3 Supported EAP scenarios and transport

RCDevs solutions provide a unified authentication framework for both users and computers, ensuring secure access control across your organization.

1. **User Authentication Based on User Certificate (EAP-TLS)**: RCDevs supports EAP-TLS (Transport Layer Security) authentication, allowing users to authenticate using digital certificates. This method enhances security by validating user identity through their unique certificates.
2. **User Authentication Based on LDAP Credentials and optionally MFA (EAP-TTLS)**: RCDevs supports EAP-TTLS for user authentication. Additionally, you can enhance security by implementing Multi-Factor Authentication (MFA) in this mode. The asked credentials during the authentication will depend on what is configured in your authentication policy(ies) configured at the WebADM level.
3. **Computer Authentication Based on Client Certificate (EAP-TLS)**: RCDevs supports EAP-TLS (Transport Layer Security) authentication, allowing computers to authenticate using digital certificates. This method enhances security by validating computer identity through their unique certificates.

The protocol used between supplicant and authenticator is EAP and between the authenticator and the authentication server is RADIUS.

1.4 Custom integrations for Certificate based authentications

OpenOTP provides SOAP API methods that can be integrated wherever you want to authenticate users/computers with SSL certificates. The same SOAP method is used by RADIUS Bridge to build and forward the authentication requests to OpenOTP. The API method will be described later in that documentation.

1.5 Prerequisites

You must meet the following requirements in order to set up EAP authentications:

- **Minimum WebADM version is v2.3.7**: Ensure that WebADM is installed at version 2.3.7 or higher. This version of WebADM is necessary to provide the infrastructure and support for computer certificate-based authentication.
- **Minimum OpenOTP version is 2.2.9**: OpenOTP is an integral part of the authentication process. Verify that OpenOTP is installed and configured, and it should be at version 2.2.9 or higher to ensure compatibility with the other components.
- **Minimum RADIUS Bridge version is 1.3.32**: The RADIUS Bridge plays a crucial role in integrating EAP authentication with RADIUS-based network access control. Ensure that the RADIUS Bridge is installed and configured correctly, and it should be running version 1.3.32 or higher to maintain compatibility with WebADM and OpenOTP.

2. Radius Bridge configuration for EAP (Authentication Server)

2.1 Radius Server configuration for EAP-TLS support

We are not explaining the Radius Bridge setup in that documentation. If Radius Bridge component is not installed and configured with your WebADM, please refer to the Radius Bridge documentation.

In order to enable the certificate based authentication feature of Radius Bridge there is 2 settings that you need to enable according to your needs:

- For user certificate based authentication, you have to enable the following setting in `/opt/radiusd/conf/radiusd.conf` located at the end of the configuration file:

```
cert_support = yes
```

That setting is mandatory to enable certificate based authentication for both users and computers.

- For computer certificate based authentication, you have to also enable the following setting in `/opt/radiusd/conf/radiusd.conf` located at the end of the configuration file:

```
machine_cert = yes
```

If not enabled, you will not be able to authenticate client certificates stored in the SQL.

Once the settings are configured, please restart Radius Bridge service.

```
/opt/radiusd/bin/radiusd restart
```

2.2 Radius Client Configuration

On your Radius Bridge server, edit the `/opt/radiusd/conf/clients.conf` and add the RADIUS client (with IP address, port and RADIUS secret) for your equipment supporting EAP protocols (Switches, WLAN Controllers, routers…).

Example:

```
client Cisco_Catalyst_Switch {
   ipaddr        = 192.168.4.253
   secret        = my_secret
}

client Cisco_WLAN_Controller {
   ipaddr        = 192.168.4.252
   secret        = testing123
}
```

Once your EAP clients are configured, please restart Radius Bridge service.

```
/opt/radiusd/bin/radiusd restart
```

3. Authenticator configuration examples

3.1 WLAN Cisco Controller

In that example, we use a Cisco WLAN controller.

The step is to configure wireless to use WPA2 Enterprise security mode and to define the RADIUS server as the authentication server for your WLAN. The specific configuration depends on the brand and model of your WLAN equipment. Please refer to your provider documentation for that part.

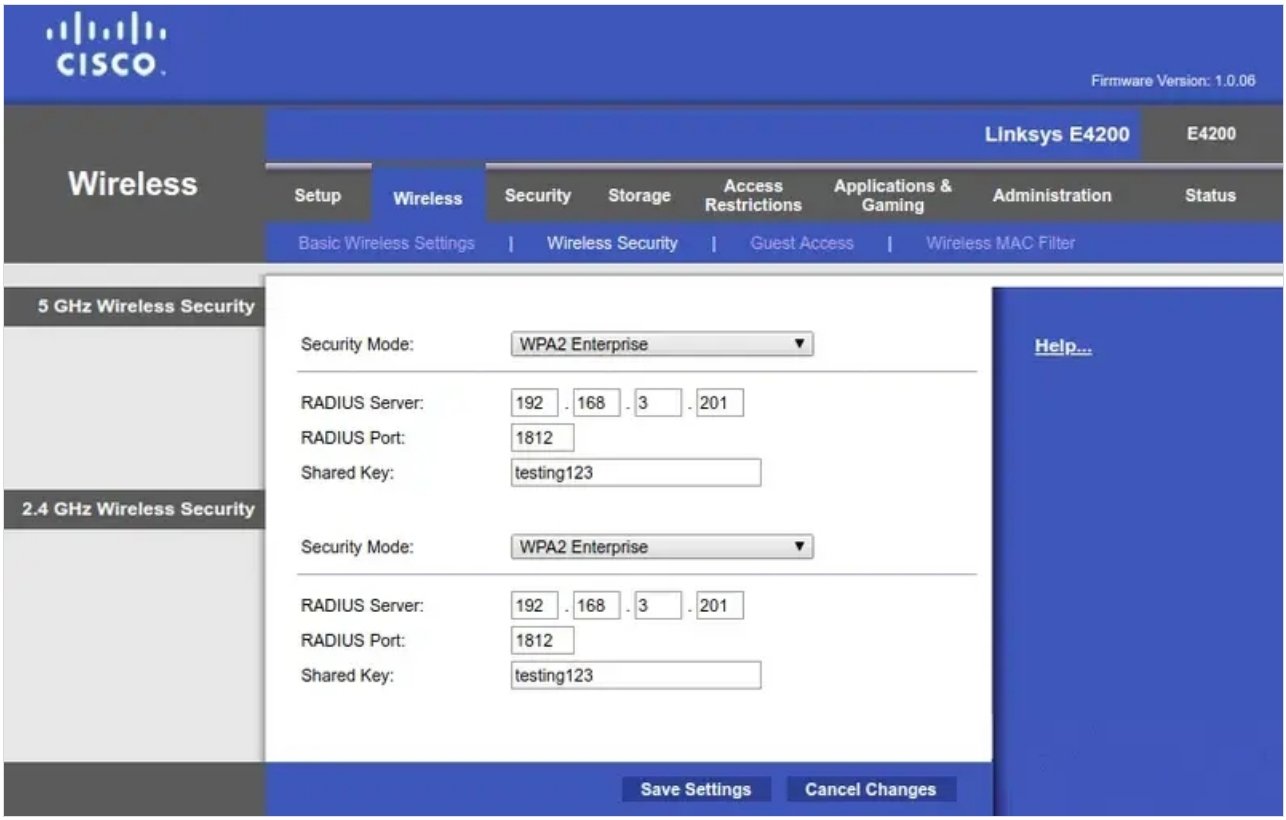
We must add a RADIUS AAA Server configuration to your Cisco WLAN controller:

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” sub tab.
4. Select the RADIUS server you configured as the “Authentication server”.
5. Click Apply to save your configuration.

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



3.2 For WLAN Access Point

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

3.3 Cisco Switch Catalyst

To configure 802.1X on my Switch, I followed the Cisco documentation. Please refer to your provider documentation to setup port based authentication (802.1X) on your switches.

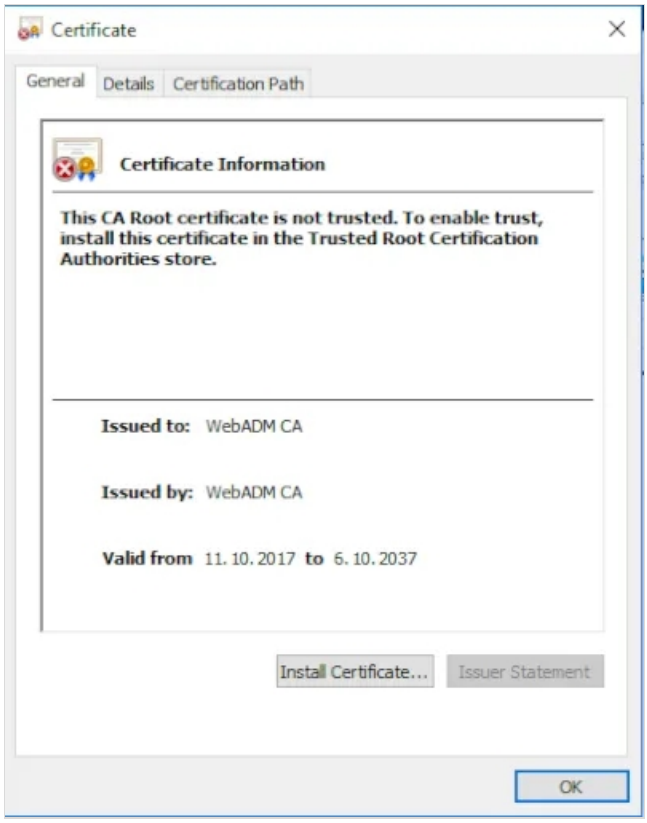
4. Supplicants configurations

4.1 Windows 10/11

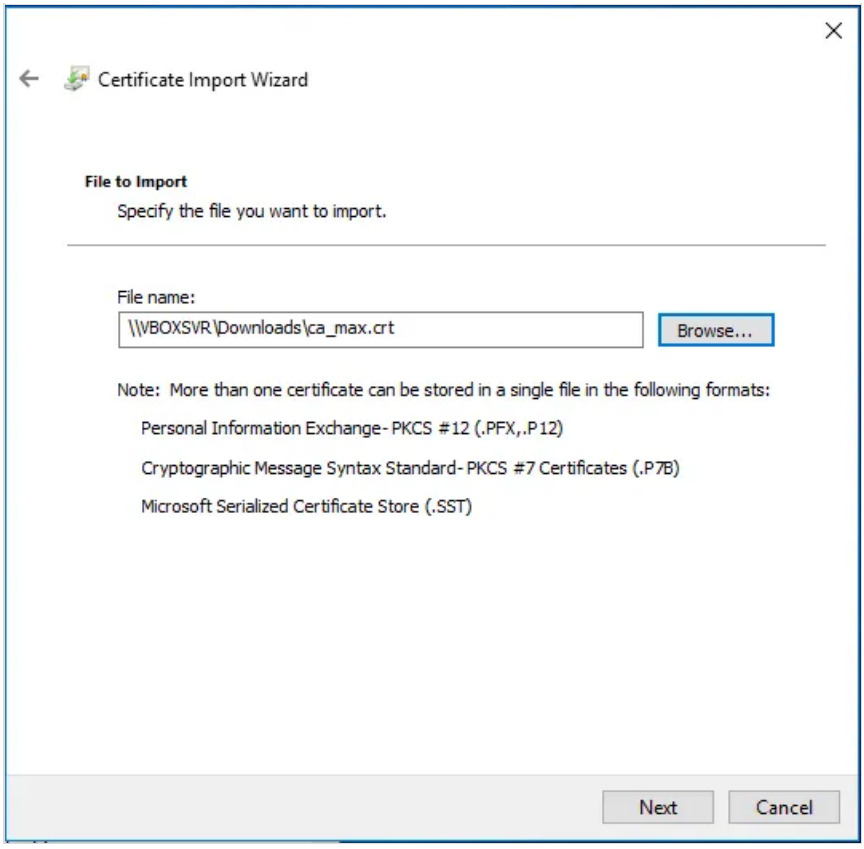
4.1.1 User certificate based authentication

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

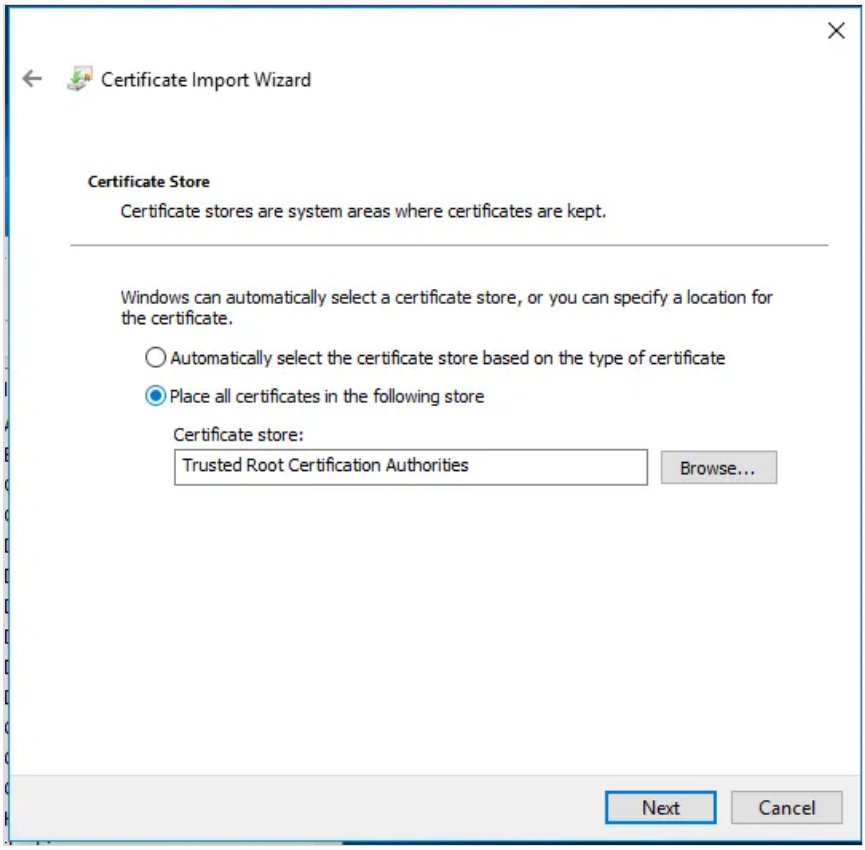
First, we must install the CA certificate of your WebADM on the Windows client. Open the CA certificate in Windows and click `Install Certificate`.



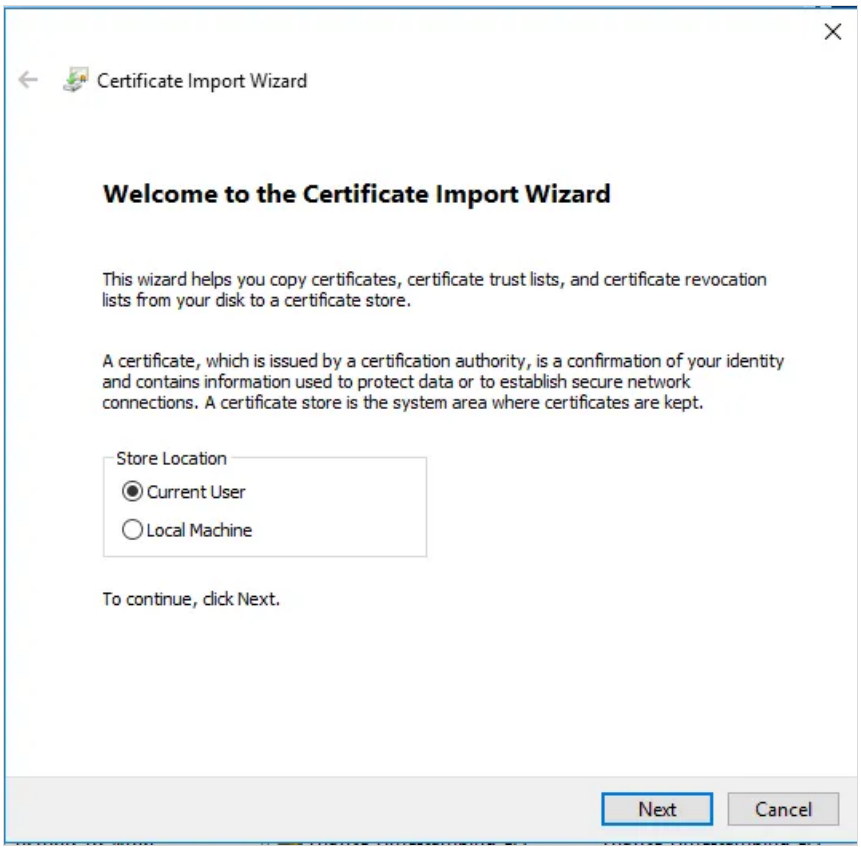
Click `Next` on the following page in Certificate Import Wizard.



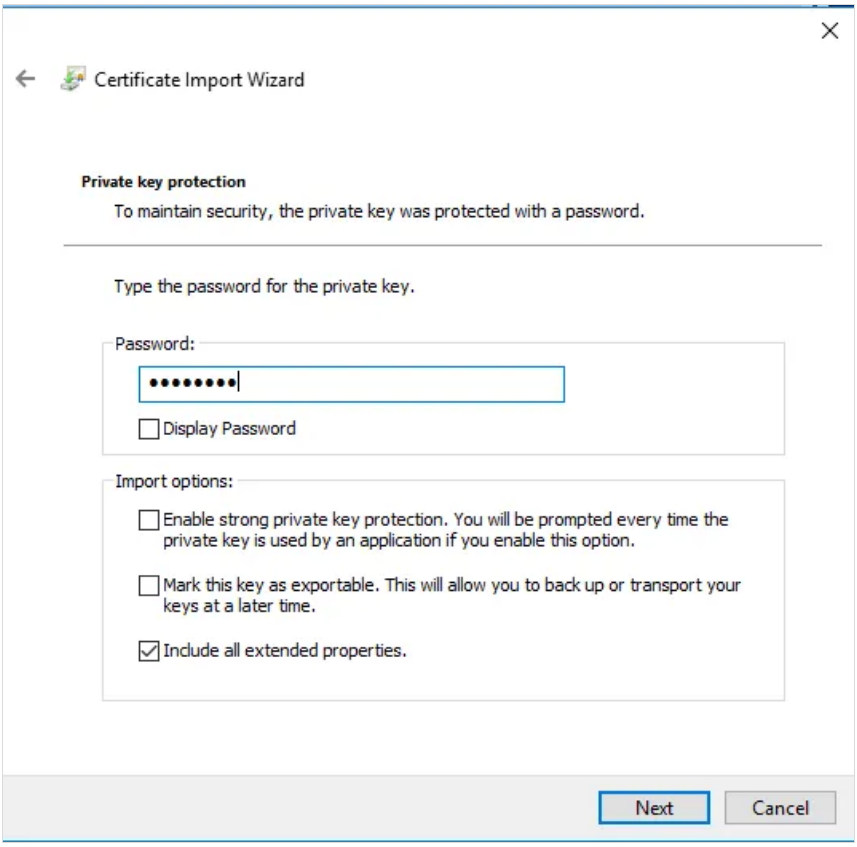
On the next page, select the certificate store in which the certificate should be installed. You must install the certificate in the "Trusted Root Certification Authorities". Click `Next` followed by `Finish` on the next page.



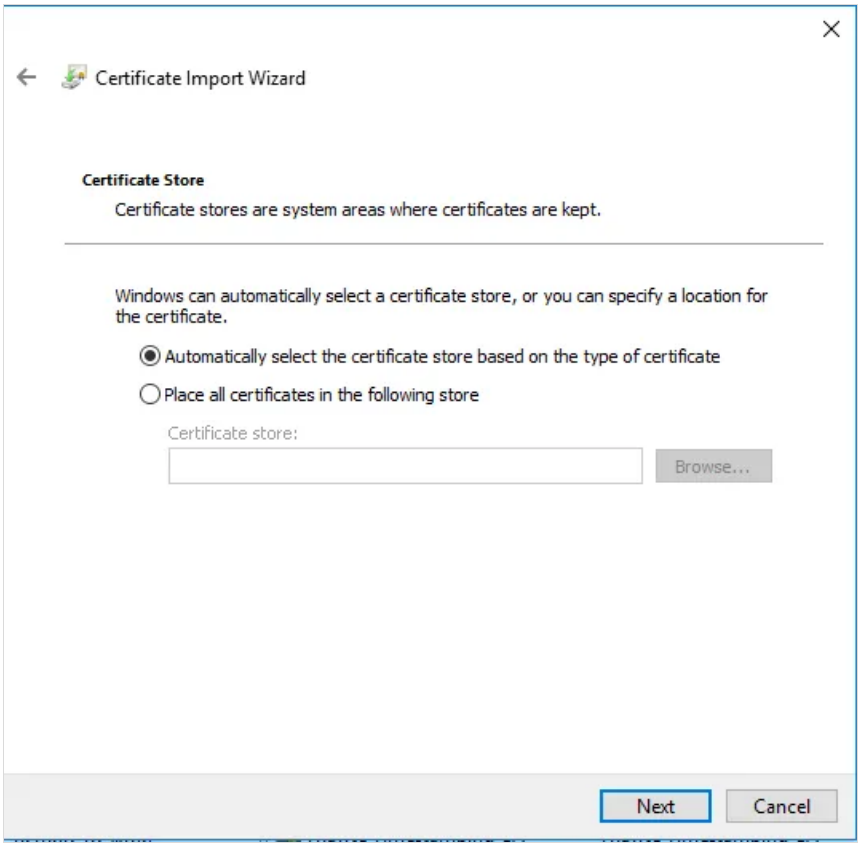
Next, we install the user certificate downloaded from Self-Service to the Windows client. Open the user certificate, select “Current User” in the wizard and click “Next” two times.



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.

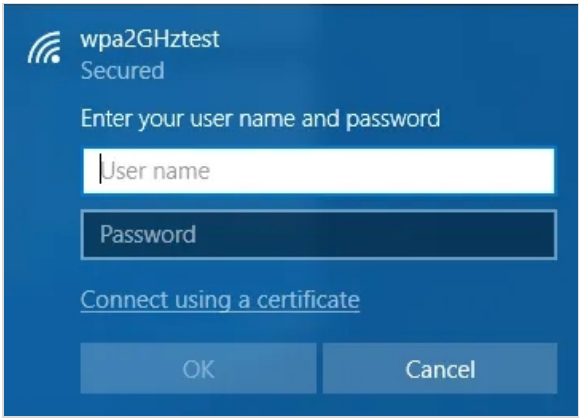


You can let Windows select the certificate store for the user certificate automatically. Click “Next” followed by `Finish`.



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”.

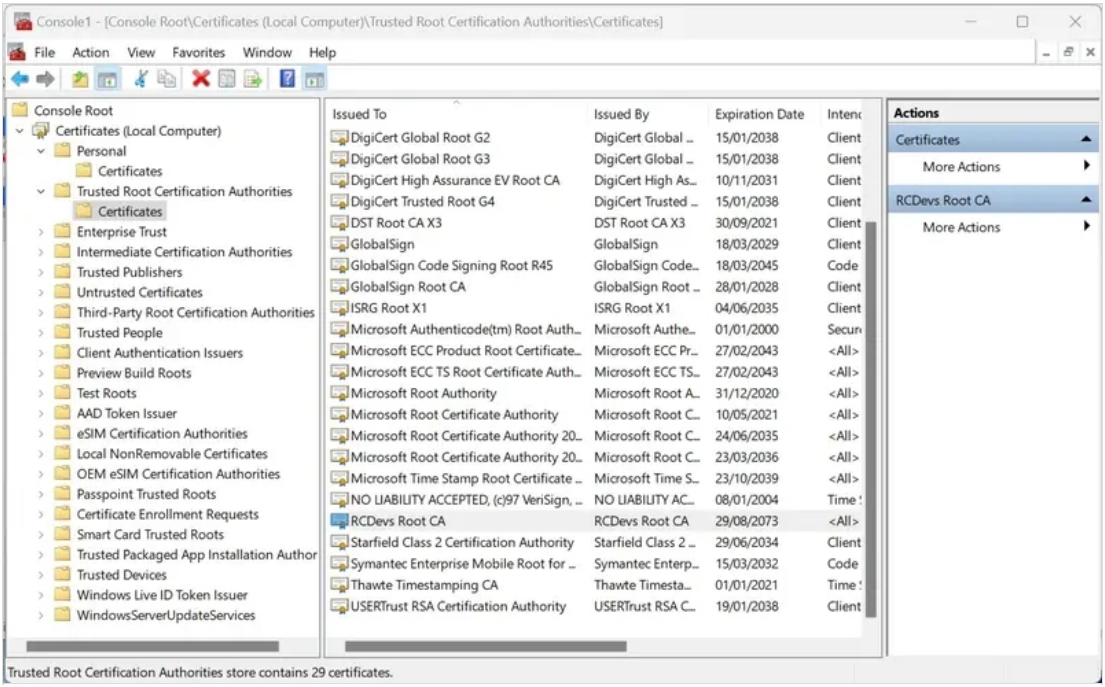


Choose the user certificate you’ve installed previously, click “OK” followed by “Connect”.

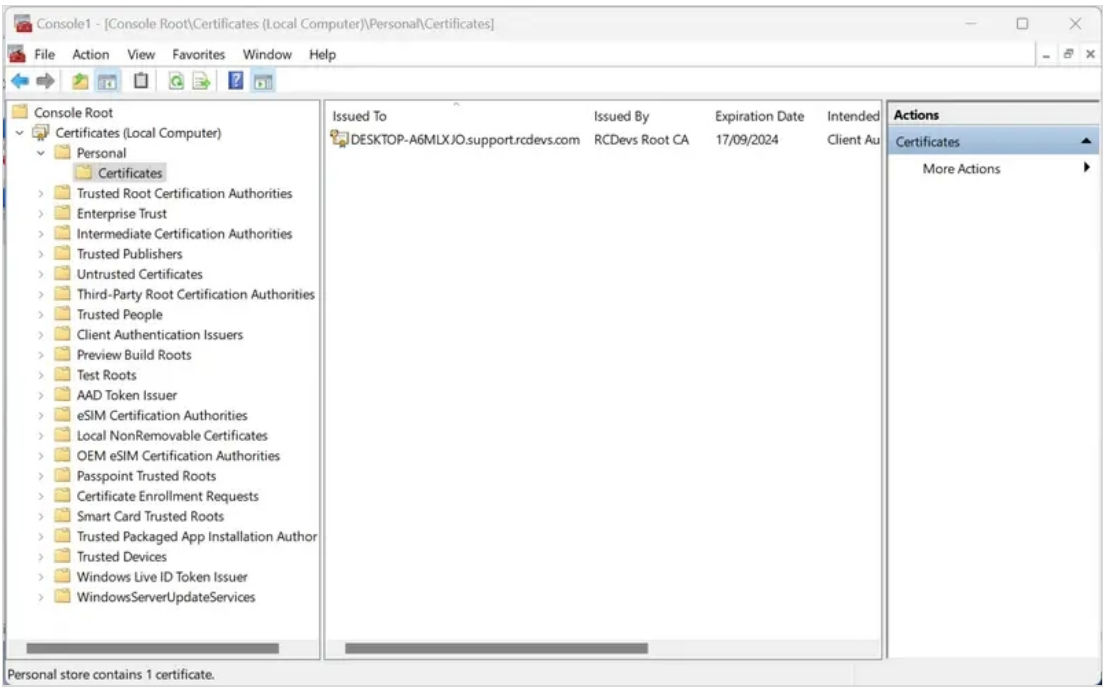
4.1.2 Device certificate based authentication

You can copy the p12 bundle previously created on your Windows machine. You also have to Trust the CA certificate of WebADM on your Windows machine. The CA certificate can be downloaded at `https://webadm_server_address/cacert`

You must add the CA certificate to the `Trusted Root Certification Authorities` in the computer store.

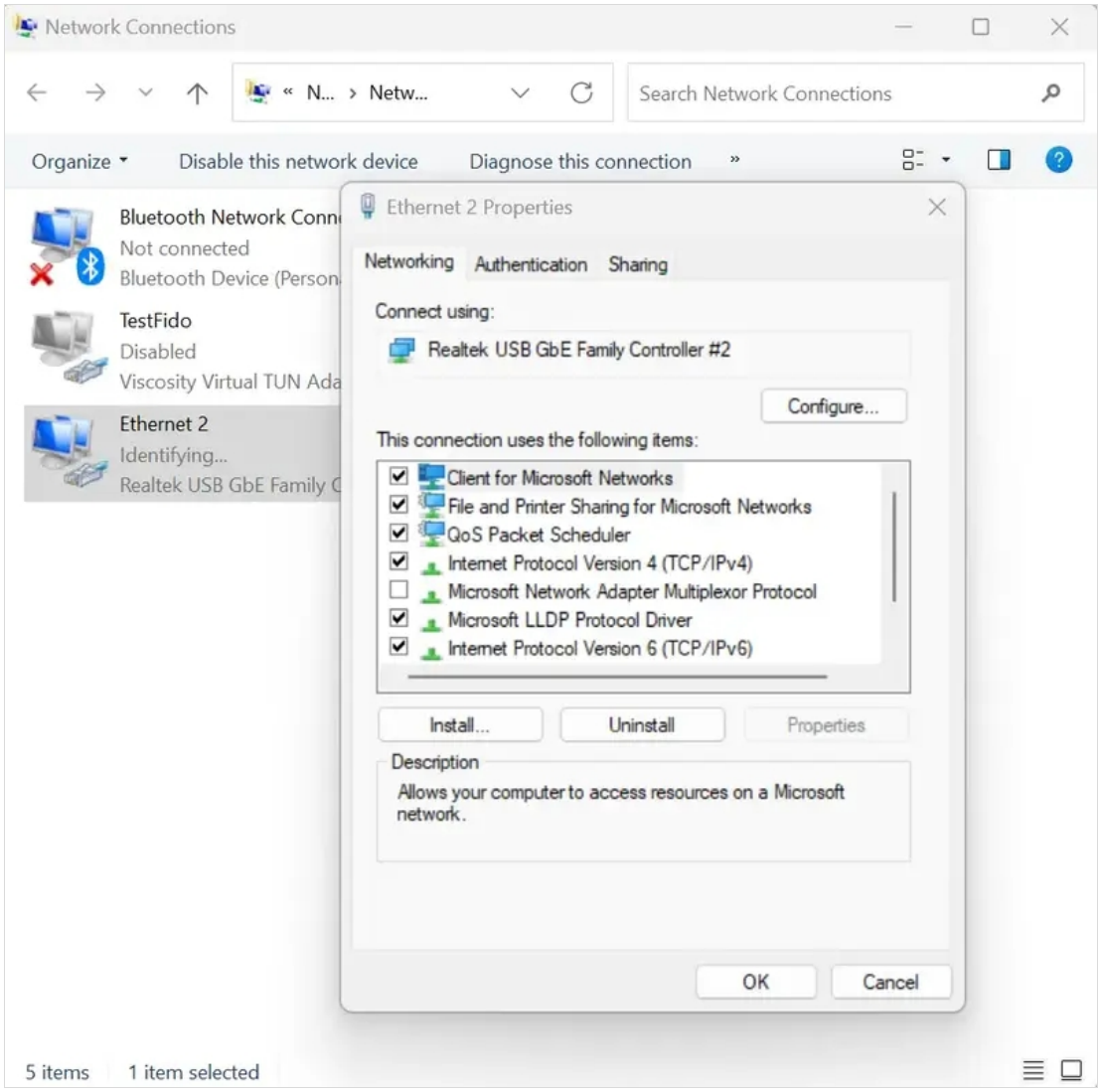


The client certificate (p12 bundle) must be installed in the `Personal` folder within the computer store. During the import process, you will be prompted to enter the password you set during the client certificate generation. For security reasons, you should also take measures to prevent the private key from being exported to minimize the risk of the certificate being reused on another machine.

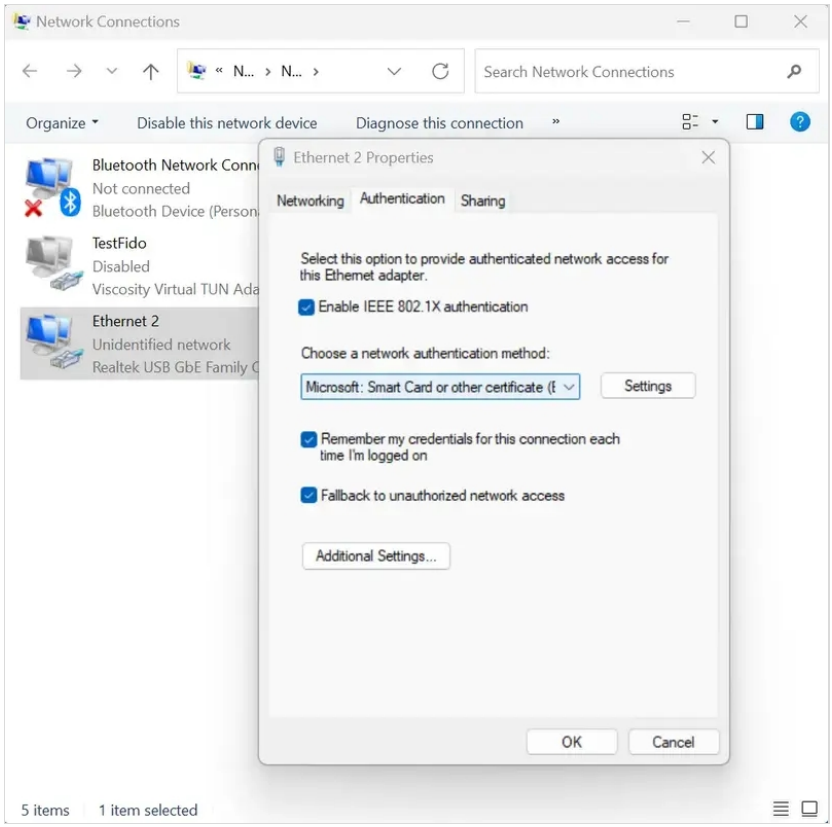


Once the certificate import is done, you can configure your network interface to set up the 802.1x authentication.

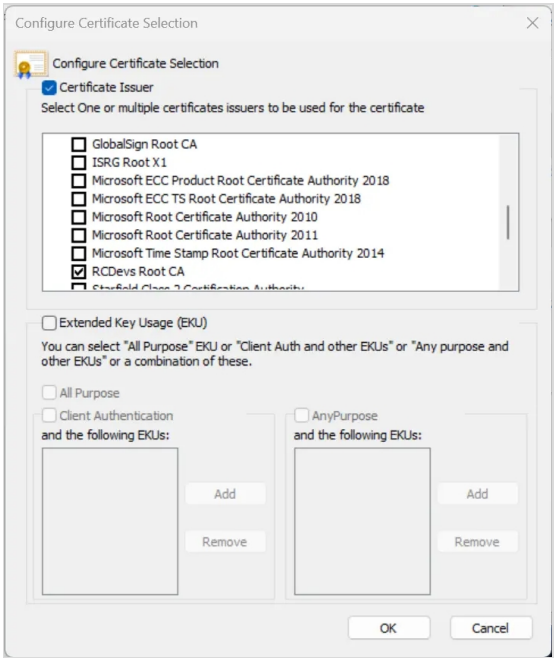
Access the `Control Panel` menu then click on `Network Connections`. Right-click on the Ethernet adapter that you want to configure then click `Properties`.



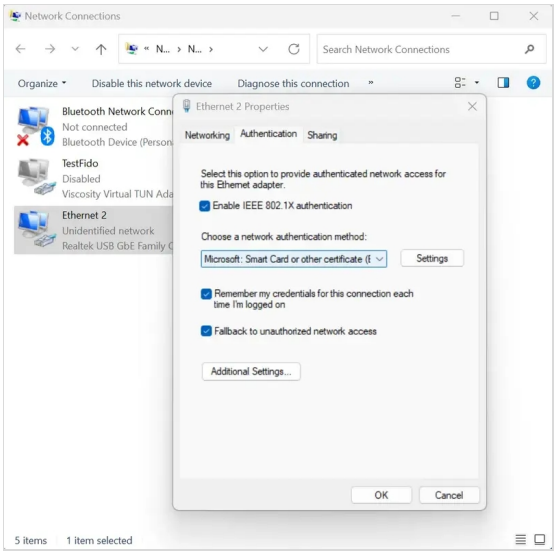
In `Properties`, click `Authentication` tab, enable the checkbox `IEEE 802.1X Authentication` and choose the network authentication method to `Microsoft: Smartcard or other Certificate`. Click then the `Settings` button.



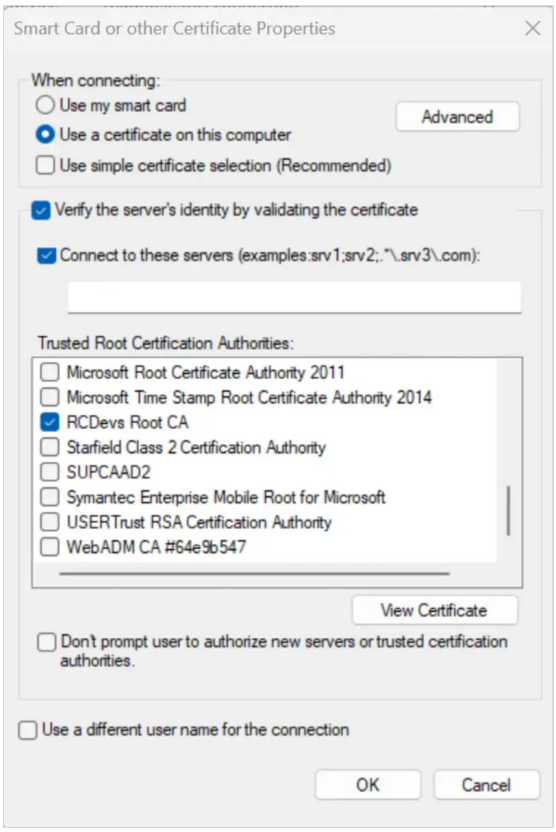
Once you are on the Configure Certificate selection view, enable the checkbox `Certificate Issuer` and you can choose the WebADM CA certificate previously imported in the `Trusted Root Certification Authorities`. Click `Ok`.



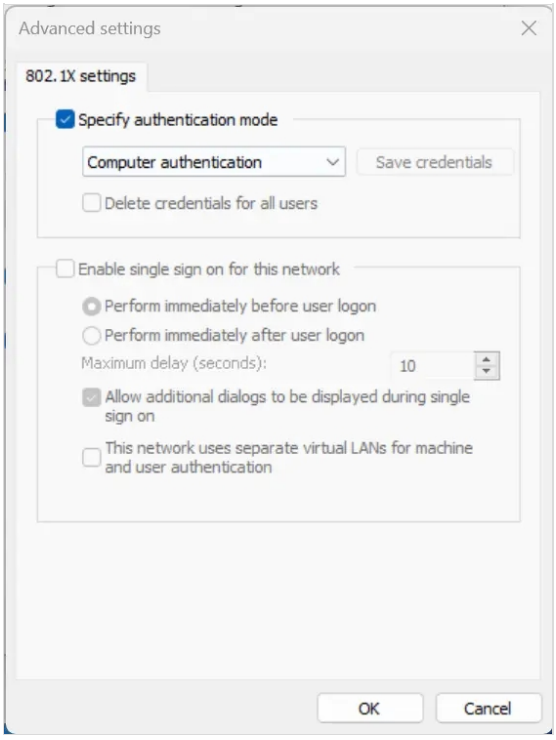
You are back to the previous page, click now the `Additional Settings` button.



You are redirected to the following page:



As we imported the certificate in the Computer Personal Certificate Store, I enabled the option `Use a certificate from this computer.` If the certificate is uploaded on a smartcard, then keep `Use my smart card` and the smartcard will have to be connected to the computer in order to use the certificate stored on it. Click then on `Advanced` button and choose the authentication mode to `Computer Authentication`




```
(0) Received Access-Request Id 173 from 192.168.4.253:1812 to 192.168.4.21:1812 length 193
(0)   NAS-IP-Address = 192.168.4.253
(0)   NAS-Port = 50002
(0)   NAS-Port-Type = Ethernet
(0)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(0)   Called-Station-Id = "00-21-1C-EA-37-42"
(0)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(0)   Service-Type = Framed-User
(0)   Framed-MTU = 1500
(0)   EAP-Message =
0x0200002c01686f73742f4445534b544f502d41364d4c584a4f2e737570706f72742e7263646576732e636f6

(0)   Message-Authenticator = 0x2ac5b9393ab7c8d5d914660863a56a72
(0) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(0)   authorize {
(0) eap: Peer sent EAP Response (code 2) ID 0 length 44
(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 = 0x2a019a2d2a008f61
(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 173 from 192.168.4.21:1812 to 192.168.4.253:1812 length 64
(0)   EAP-Message = 0x010100061520
(0)   Message-Authenticator = 0x00000000000000000000000000000000
(0)   State = 0x2a019a2d2a008f6172f171851926dbf8
(0) Finished request
Waking up in 9.9 seconds.
(1) Received Access-Request Id 174 from 192.168.4.253:1812 to 192.168.4.21:1812 length 173
(1)   NAS-IP-Address = 192.168.4.253
(1)   NAS-Port = 50002
(1)   NAS-Port-Type = Ethernet
(1)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(1)   Called-Station-Id = "00-21-1C-EA-37-42"
(1)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(1)   Service-Type = Framed-User
(1)   Framed-MTU = 1500
(1)   State = 0x2a019a2d2a008f6172f171851926dbf8
(1)   EAP-Message = 0x02010006030d
(1)   Message-Authenticator = 0x298794c7ffc1e7c59739b04ca20727dc
(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 0x2a019a2d2a008f61
(1) eap: Finished EAP session with state 0x2a019a2d2a008f61
(1) eap: Previous EAP request found for state 0x2a019a2d2a008f61, 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 = 0x2a019a2d2b039761
(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 174 from 192.168.4.21:1812 to 192.168.4.253:1812 length 64
```

```
(1)   EAP-Message = 0x010200060d20
(1)   Message-Authenticator = 0x00000000000000000000000000000000
(1)   State = 0x2a019a2d2b03976172f171851926dbf8
(1) Finished request
Waking up in 9.9 seconds.
(2) Received Access-Request Id 175 from 192.168.4.253:1812 to 192.168.4.21:1812 length 430
(2)   NAS-IP-Address = 192.168.4.253
(2)   NAS-Port = 50002
(2)   NAS-Port-Type = Ethernet
(2)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(2)   Called-Station-Id = "00-21-1C-EA-37-42"
(2)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(2)   Service-Type = Framed-User
(2)   Framed-MTU = 1500
(2)   State = 0x2a019a2d2b03976172f171851926dbf8
(2)   EAP-Message =
0x020201050d80000000fb16030100f6010000f2030386a0bc99a2f21bee48f1aa5956c7f89efff335f808418dd

(2)   Message-Authenticator = 0xb8625c14068a4f535607ebc48f9b1d5d
(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 261
(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 0x2a019a2d2b039761
(2) eap: Finished EAP session with state 0x2a019a2d2b039761
(2) eap: Previous EAP request found for state 0x2a019a2d2b039761, 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 251 bytes
(2) eap_tls: (TLS) EAP Got all data (251 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) 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 = 0x2a019a2d28029761
(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)   TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(2)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(2)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(2)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(2)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(2)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(2) Sent Access-Challenge Id 175 from 192.168.4.21:1812 to 192.168.4.253:1812 length 1068
(2)   EAP-Message =
0x010303ec0dc000000cb316030300350200003103030 2d214a8f835dfda6fe8933ad05921668c24f6ce99a81

(2)   Message-Authenticator = 0x00000000000000000000000000000000
(2)   State = 0x2a019a2d2802976172f171851926dbf8
(2) Finished request
Waking up in 9.9 seconds.
(3) Received Access-Request Id 176 from 192.168.4.253:1812 to 192.168.4.21:1812 length 173
(3)   NAS-IP-Address = 192.168.4.253
(3)   NAS-Port = 50002
(3)   NAS-Port-Type = Ethernet
(3)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(3)   Called-Station-Id = "00-21-1C-EA-37-42"
(3)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(3)   Service-Type = Framed-User
(3)   Framed-MTU = 1500
(3)   State = 0x2a019a2d2802976172f171851926dbf8
(3)   EAP-Message = 0x020300060d00
(3)   Message-Authenticator = 0x05115c60ec77564bc18f7ae8bb79695f
(3) Restoring &session-state
(3)   &session-state:Framed-MTU = 994
(3)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(3)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(3)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(3)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(3)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(3)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(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 0x2a019a2d28029761
(3) eap: Finished EAP session with state 0x2a019a2d28029761
(3) eap: Previous EAP request found for state 0x2a019a2d28029761, 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 = 0x2a019a2d29059761
(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)   TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(3)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(3)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(3)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(3)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(3)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(3) Sent Access-Challenge Id 176 from 192.168.4.21:1812 to 192.168.4.253:1812 length 1068
(3)   EAP-Message =
0x010403ec0dc000000cb365e0f50616561bd3e6513de7f3d235c02f1b51e30e2d3ab6bb527998a88654452c9

(3)   Message-Authenticator = 0x00000000000000000000000000000000
(3)   State = 0x2a019a2d2905976172f171851926dbf8
(3) Finished request
Waking up in 9.9 seconds.
(4) Received Access-Request Id 177 from 192.168.4.253:1812 to 192.168.4.21:1812 length 173
(4)   NAS-IP-Address = 192.168.4.253
(4)   NAS-Port = 50002
(4)   NAS-Port-Type = Ethernet
(4)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(4)   Called-Station-Id = "00-21-1C-EA-37-42"
(4)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(4)   Service-Type = Framed-User
(4)   Framed-MTU = 1500
(4)   State = 0x2a019a2d2905976172f171851926dbf8
(4)   EAP-Message = 0x020400060d00
(4)   Message-Authenticator = 0x59860564389662c6f35e69fcc49af4ab
(4) Restoring &session-state
(4)   &session-state:Framed-MTU = 994
(4)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(4)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
```

```
(4)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(4)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(4)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(4)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(4)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(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 0x2a019a2d29059761
(4) eap: Finished EAP session with state 0x2a019a2d29059761
(4) eap: Previous EAP request found for state 0x2a019a2d29059761, 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 1004
(4) eap: EAP session adding &reply:State = 0x2a019a2d2e049761
(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)   TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(4)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(4)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(4)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(4)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(4)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(4) Sent Access-Challenge Id 177 from 192.168.4.21:1812 to 192.168.4.253:1812 length 1068
(4)   EAP-Message =
0x010503ec0dc000000cb30f002062e83091a12eb1cb6c0b967dd3b1b83533e17ddd8ab58bdd705e30150f3d

(4)   Message-Authenticator = 0x00000000000000000000000000000000
(4)   State = 0x2a019a2d2e04976172f171851926dbf8
(4) Finished request
Waking up in 9.9 seconds.
(5) Received Access-Request Id 178 from 192.168.4.253:1812 to 192.168.4.21:1812 length 173
(5)   NAS-IP-Address = 192.168.4.253
(5)   NAS-Port = 50002
(5)   NAS-Port-Type = Ethernet
(5)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(5)   Called-Station-Id = "00-21-1C-EA-37-42"
(5)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(5)   Service-Type = Framed-User
```

```
(5)   Framed-MTU = 1500
(5)   State = 0x2a019a2d2e04976172f171851926dbf8
(5)   EAP-Message = 0x020500060d00
(5)   Message-Authenticator = 0x21398d5fd30706f8857b51fd3ee7ea0d
(5) Restoring &session-state
(5)   &session-state:Framed-MTU = 994
(5)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(5)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(5)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(5)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(5)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(5)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(5) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(5)   authorize {
(5) eap: Peer sent EAP Response (code 2) ID 5 length 6
(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 0x2a019a2d2e049761
(5) eap: Finished EAP session with state 0x2a019a2d2e049761
(5) eap: Previous EAP request found for state 0x2a019a2d2e049761, 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) Peer ACKed our handshake fragment
(5) eap: Sending EAP Request (code 1) ID 6 length 279
(5) eap: EAP session adding &reply:State = 0x2a019a2d2f079761
(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)   TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(5)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(5)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(5)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(5)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(5)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(5) Sent Access-Challenge Id 178 from 192.168.4.21:1812 to 192.168.4.253:1812 length 339
(5)   EAP-Message =
0x010601170d8000000cb3e5c5a1a7ee59b56934c4592faba1840501006c799ebd2af51f07c58bbcec8894627

(5)   Message-Authenticator = 0x00000000000000000000000000000000
(5)   State = 0x2a019a2d2f07976172f171851926dbf8
(5) Finished request
Waking up in 9.9 seconds.
(6) Received Access-Request Id 179 from 192.168.4.253:1812 to 192.168.4.21:1812 length 1669
```

```
(6) Received Access-Request Id 179 from 192.168.4.253:50002 to 192.168.4.211:1812 length 1009
(6)   NAS-IP-Address = 192.168.4.253
(6)   NAS-Port = 50002
(6)   NAS-Port-Type = Ethernet
(6)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(6)   Called-Station-Id = "00-21-1C-EA-37-42"
(6)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(6)   Service-Type = Framed-User
(6)   Framed-MTU = 1500
(6)   State = 0x2a019a2d2f07976172f171851926dbf8
(6)   EAP-Message =
0x020605d40dc0000006c6160303068e0b00053c000539000536308205323082031aa003020102021100d9c

(6)   Message-Authenticator = 0x5caf5e6d061691584c9093801846c341
(6) Restoring &session-state
(6)   &session-state:Framed-MTU = 994
(6)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(6)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(6)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(6)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(6)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(6)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(6) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(6)   authorize {
(6) eap: Peer sent EAP Response (code 2) ID 6 length 1492
(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 0x2a019a2d2f079761
(6) eap: Finished EAP session with state 0x2a019a2d2f079761
(6) eap: Previous EAP request found for state 0x2a019a2d2f079761, 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 Peer says that the final record size will be 1734 bytes
(6) eap_tls: (TLS) EAP Expecting 2 fragments
(6) eap_tls: (TLS) EAP Got first TLS fragment (1482 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 = 0x2a019a2d2c069761
(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)   TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(6)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
```

```
(6)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(6)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(6)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(6)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(6) Sent Access-Challenge Id 179 from 192.168.4.21:1812 to 192.168.4.253:1812 length 64
(6)   EAP-Message = 0x010700060d00
(6)   Message-Authenticator = 0x00000000000000000000000000000000
(6)   State = 0x2a019a2d2c06976172f171851926dbf8
(6) Finished request
Waking up in 9.8 seconds.
(7) Received Access-Request Id 180 from 192.168.4.253:1812 to 192.168.4.21:1812 length 427
(7)   NAS-IP-Address = 192.168.4.253
(7)   NAS-Port = 50002
(7)   NAS-Port-Type = Ethernet
(7)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(7)   Called-Station-Id = "00-21-1C-EA-37-42"
(7)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(7)   Service-Type = Framed-User
(7)   Framed-MTU = 1500
(7)   State = 0x2a019a2d2c06976172f171851926dbf8
(7)   EAP-Message =
0x020701020d00e20d42035e93afe672c521d5bb19182b78d8a423c7e3c4ce4c8b84b0445c61a4b95dbc602b

(7)   Message-Authenticator = 0xd2334d833b7fb6a9701c3c158f17a322
(7) Restoring &session-state
(7)   &session-state:Framed-MTU = 994
(7)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(7)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(7)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(7)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(7)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(7)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(7) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(7)   authorize {
(7) eap: Peer sent EAP Response (code 2) ID 7 length 258
(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 0x2a019a2d2c069761
(7) eap: Finished EAP session with state 0x2a019a2d2c069761
(7) eap: Previous EAP request found for state 0x2a019a2d2c069761, 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 (252 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) recv TLS 1.2 Handshake, Certificate
(7) eap_tls: (TLS) Creating attributes from server certificate
(7) eap_tls:   TLS-Cert-Serial := "5a42eea11d8d1528cd307d761cc682ec4da6aab3"
(7) eap_tls:   TLS-Cert-Expiration := "20730829094146Z"
(7) eap_tls:   TLS-Cert-Valid-Since := "230911094146Z"
(7) eap_tls:   TLS-Cert-Subject := "/CN=RCDevs Root CA/OU=CA/O=RCDevs Support/C=LU"
(7) eap_tls:   TLS-Cert-Issuer := "/CN=RCDevs Root CA/OU=CA/O=RCDevs Support/C=LU"
(7) eap_tls:   TLS-Cert-Common-Name := "RCDevs Root CA"
(7) eap_tls: (TLS) Creating attributes from client certificate
(7) eap_tls:   TLS-Client-Cert-Serial := "d9cd10deb891f2698216842aaae5cc53"
(7) eap_tls:   TLS-Client-Cert-Expiration := "240917160009Z"
(7) eap_tls:   TLS-Client-Cert-Valid-Since := "230918160009Z"
(7) eap_tls:   TLS-Client-Cert-Subject := "/CN=DESKTOP-
A6MLXJO.support.rcdevs.com/description=CLIENT/O=RCDevs Support/organizationIdentifier=VATLU-
00000000"
(7) eap_tls:   TLS-Client-Cert-Issuer := "/CN=RCDevs Root CA/OU=CA/O=RCDevs Support/C=LU"
(7) eap_tls:   TLS-Client-Cert-Common-Name := "DESKTOP-A6MLXJO.support.rcdevs.com"
(7) eap_tls:   TLS-Client-Cert-Subject-Alt-Name-Dns := "DESKTOP-A6MLXJO.support.rcdevs.com"
(7) eap_tls:   TLS-Client-Cert-X509v3-Extended-Key-Usage += "TLS Web Client Authentication"
(7) eap_tls:   TLS-Client-Cert-X509v3-Extended-Key-Usage-OID += "1.3.6.1.5.5.7.3.2"
Certificate chain - 1 cert(s) untrusted
(TLS) untrusted certificate with depth [0] subject name /CN=DESKTOP-
A6MLXJO.support.rcdevs.com/description=CLIENT/O=RCDevs Support/organizationIdentifier=VATLU-
00000000
(7) eap_tls: Starting OCSP Request
OpenOTP PKI login succeeded
(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_tls:   TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"
(7) eap_tls:   TLS-Session-Version = "TLS 1.2"
(7) eap: Sending EAP Request (code 1) ID 8 length 61
(7) eap: EAP session adding &reply:State = 0x2a019a2d2d099761
(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)   TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(7)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(7)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(7)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(7)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(7)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(7)   TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Certificate"
(7)   TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"
(7)   TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, CertificateVerify"
(7)   TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"
(7)   TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"
(7)   TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"
(7)   TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"
(7)   TLS-Session-Version = "TLS 1.2"
(7) Sent Access-Challenge Id 180 from 192.168.4.21:1812 to 192.168.4.253:1812 length 119
(7)   EAP-Message =
0x0108003d0d8000000033140303000101160303002 85fefc6a50b9e632a6910d6d47a124bfd5b7219f97268

(7)   Message-Authenticator = 0x00000000000000000000000000000000
(7)   State = 0x2a019a2d2d09976172f171851926dbf8
(7) Finished request
Waking up in 9.8 seconds.
(8) Received Access-Request Id 181 from 192.168.4.253:1812 to 192.168.4.21:1812 length 173
(8)   NAS-IP-Address = 192.168.4.253
(8)   NAS-Port = 50002
(8)   NAS-Port-Type = Ethernet
(8)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(8)   Called-Station-Id = "00-21-1C-EA-37-42"
(8)   Calling-Station-Id = "00-13-3B-A0-43-3E"
(8)   Service-Type = Framed-User
(8)   Framed-MTU = 1500
(8)   State = 0x2a019a2d2d09976172f171851926dbf8
(8)   EAP-Message = 0x020800060d00
(8)   Message-Authenticator = 0xee463f6371044aff2ccfedf9e4ee3d8d
(8) Restoring &session-state
(8)   &session-state:Framed-MTU = 994
(8)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.3 Handshake, ClientHello"
(8)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHello"
(8)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Certificate"
(8)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerKeyExchange"
(8)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, CertificateRequest"
(8)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, ServerHelloDone"
(8)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Certificate"
(8)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, ClientKeyExchange"
(8)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, CertificateVerify"
(8)   &session-state:TLS-Session-Information = "(TLS) recv TLS 1.2 Handshake, Finished"
(8)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 ChangeCipherSpec"
(8)   &session-state:TLS-Session-Information = "(TLS) send TLS 1.2 Handshake, Finished"
(8)   &session-state:TLS-Session-Cipher-Suite = "ECDHE-RSA-AES256-GCM-SHA384"
(8)   &session-state:TLS-Session-Version = "TLS 1.2"
```

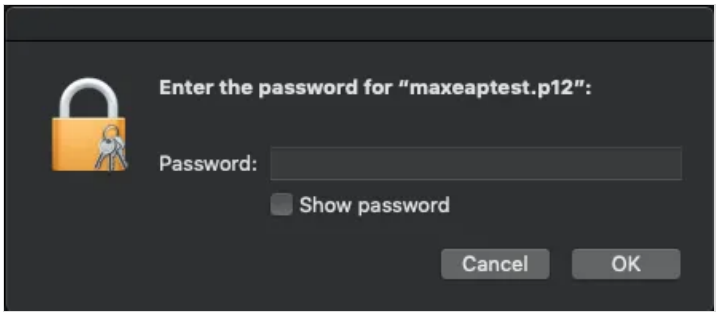
```
(8) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(8)   authorize {
(8) eap: Peer sent EAP Response (code 2) ID 8 length 6
(8) eap: Continuing tunnel setup
(8)     [eap] = ok
(8)   } # authorize = ok
(8) Found Auth-Type = EAP
(8) # Executing group from file /opt/radiusd/lib/radiusd.ini
(8)   Auth-Type EAP {
(8) eap: Expiring EAP session with state 0x2a019a2d2d099761
(8) eap: Finished EAP session with state 0x2a019a2d2d099761
(8) eap: Previous EAP request found for state 0x2a019a2d2d099761, 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
(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: [host/DESKTOP-A6MLXJO.support.rcdevs.com] (from client any port 50002 cli 00-13-3B-A0-43-3E)
(8) Sent Access-Accept Id 181 from 192.168.4.21:1812 to 192.168.4.253:1812 length 201
(8)   MS-MPPE-Recv-Key = 0xec58342777317bb961c9ef3ff7f396221acaf7b1714eb5c1fd558878ae69267b
(8)   MS-MPPE-Send-Key = 0x00af4187cd99abeb6966d121d9b4c188e529ddcfe919bf4a420242cc58ab4b08
(8)   EAP-Message = 0x03080004
(8)   Message-Authenticator = 0x00000000000000000000000000000000
(8)   User-Name = "host/DESKTOP-A6MLXJO.support.rcdevs.com"
(8) Finished request
Waking up in 9.8 seconds.
```

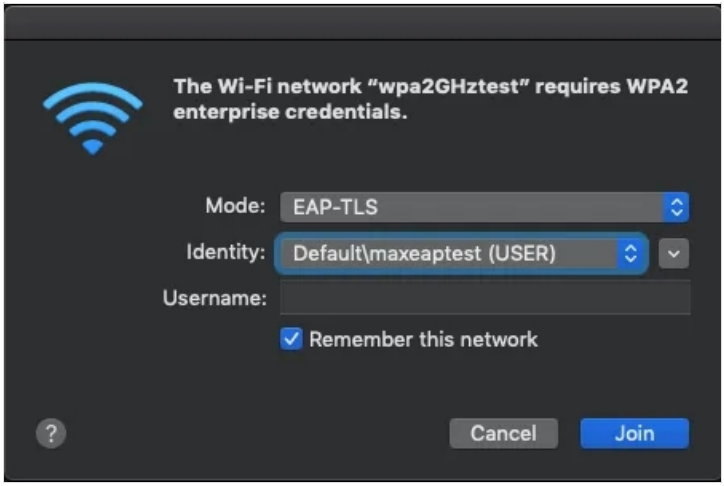
You are now authenticated on your switch.

4.2 macOS / iOS

4.2.1 User certificate based authentication

In macOS, open the downloaded user certificate to install it into the keychain. Input the password which protect the p12 bundle.





Once my certificate has been issued and imported in my keychain, 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 Wi-Fi 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 click Connect and few seconds after, you are connected to the Wi-Fi. See below, the EAP-TLS debug logs from Radius Bridge:

```
(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)   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 = 0x02000013014465666175 6c745c726f6c616e64
(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) 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 =
0x020200a10d800000009716030100920100008e030360e5e1d43c1356e4e327db12aa19c43cbf67a94d117

(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 =
0x010303ec0dc0000008ee160303003d0200003903033b6d140fdc958c73e51eb3c0cfbaea24f5211392e6b89

(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: 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 5a180f32303731303431343133303134395a30343119301

(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 =
0x020504fc0dc0000007f316030306630b00065f00065c00030a30820306308201eea003020102020125300d

(5)   Message-Authenticator = 0x0333c9763187fe93e3a60ef5a6432197
(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 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 =
0x020603070d00b7e2f7a30701ef63fcc94b0203010001a350304e301d0603551d0e0416041428a7dc1346e1

(6)   Message-Authenticator = 0x3a15445e85a724b07397c16d6dcda6bd
(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 8f3c248d2faf970c644015b2a2588a7eea20b7925835a

(6)   Message-Authenticator = 0x00000000000000000000000000000000
(6)   State = 0xd85aceebde5dc318ad985fadddd9ba17
(6) Finished request
Waking up in 9.9 seconds
```

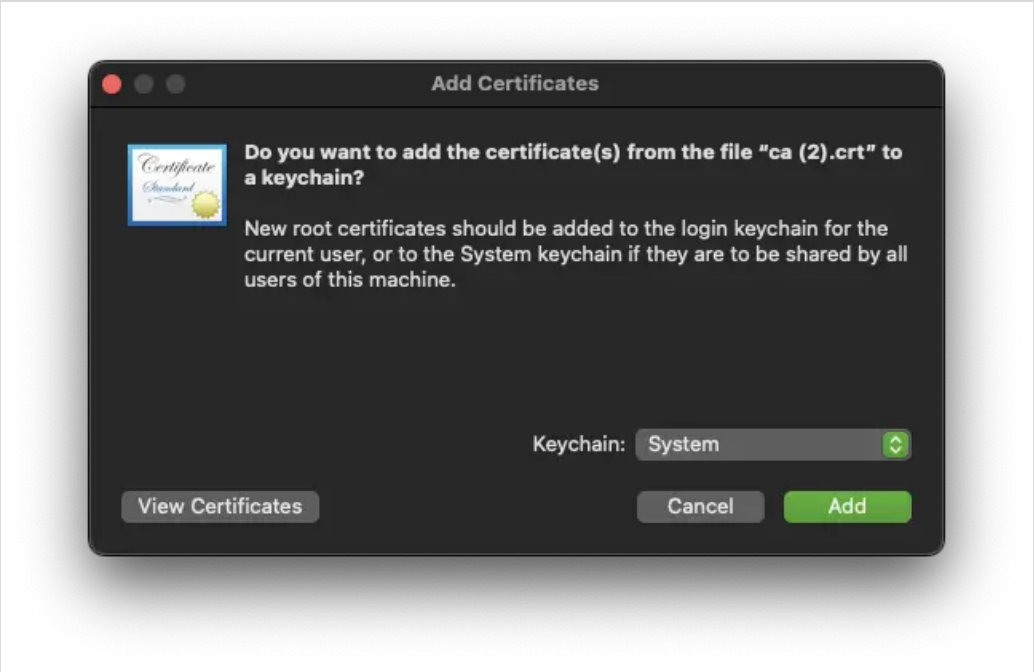
```
Waking up in 9.9 seconds.
(6) Cleaning up request packet ID 0 with timestamp +749
(7) Received Access-Request Id 0 from 192.168.4.250:32768 to 192.168.4.20:1812 length 146
(7)   User-Name = "Default\\roland"
(7)   NAS-IP-Address = 192.168.4.250
(7)   Called-Station-Id = "586d8fa0308d"
(7)   Calling-Station-Id = "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"
```

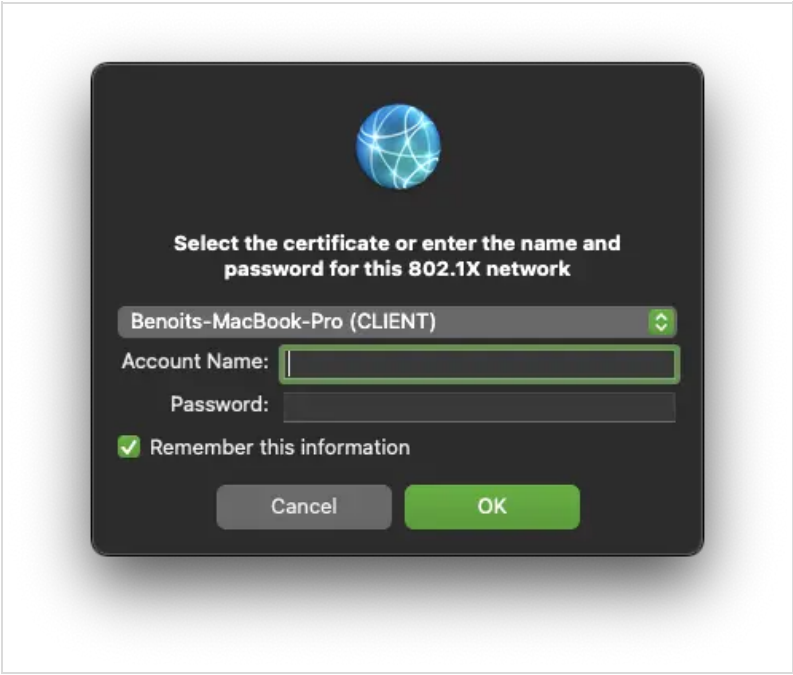
4.2.2 Device certificate based authentication

You can copy the p12 bundle previously created on your macOS machine. The CA certificate of WebADM will be needed during for the connection setup and can be downloaded at `https://webadm_server_address/cacert`. You must add the CA certificate of WebADM into the `System Keychain` and the p12 bundle into the `System Keychain` in order to be shared between the different users available on the same machine.

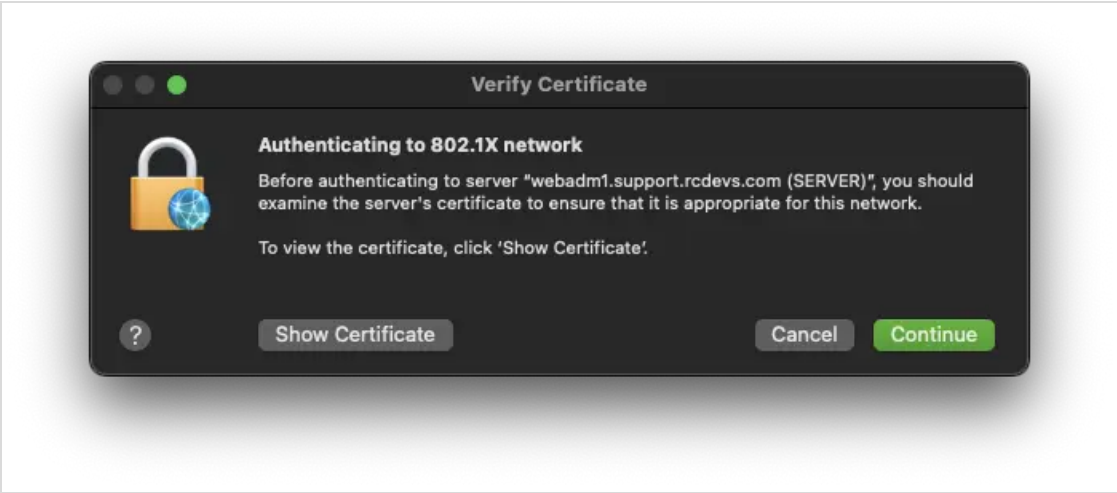


Provide the password which protect the p12 bundle and click Ok:

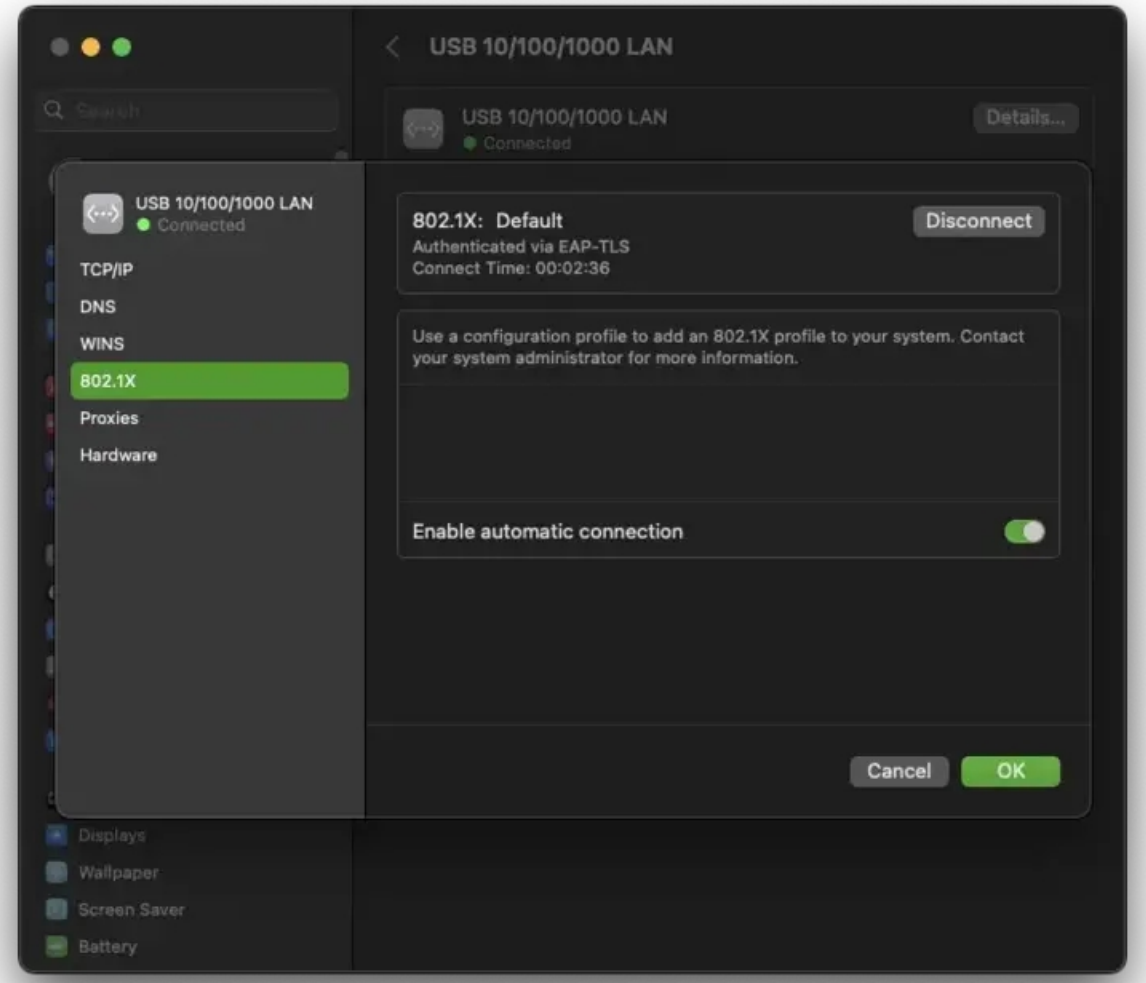
When you will connect the ethernet cable to your OSX device, the system will prompt you to choose the client certificate. Select the client certificate previously imported. Keep account name and password empty as we are not authenticating the user but the client machine and then press `Ok`:



You may be prompted to trust the certificate of the authentication server as below:



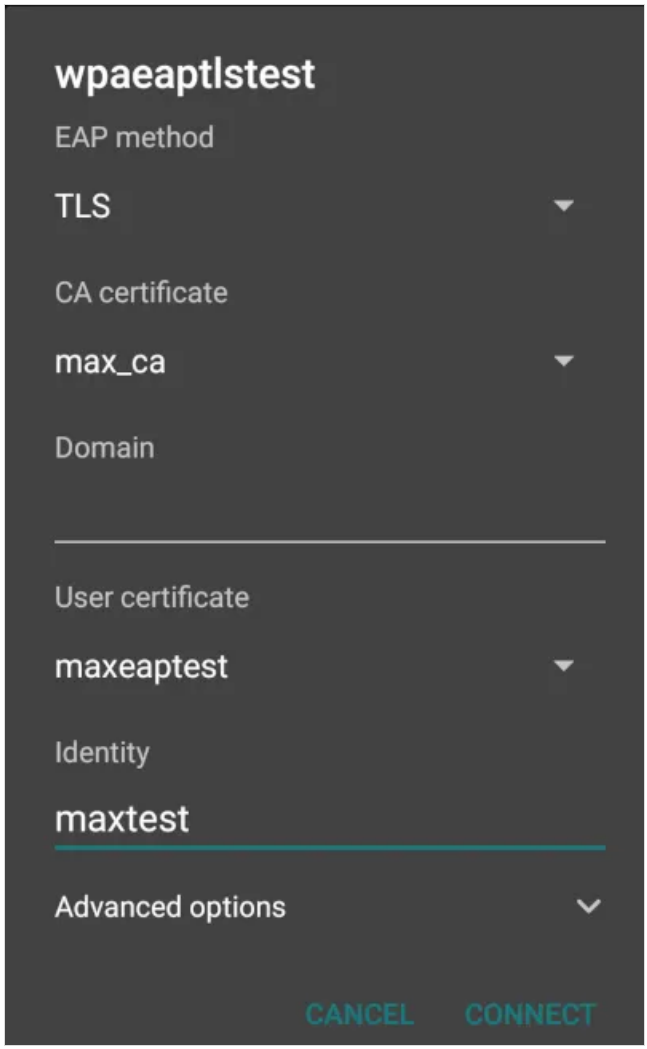
Press `Continue` and provide the OSX credentials to access the keychain.



4.3 Android

4.3.1 User certificate based authentication

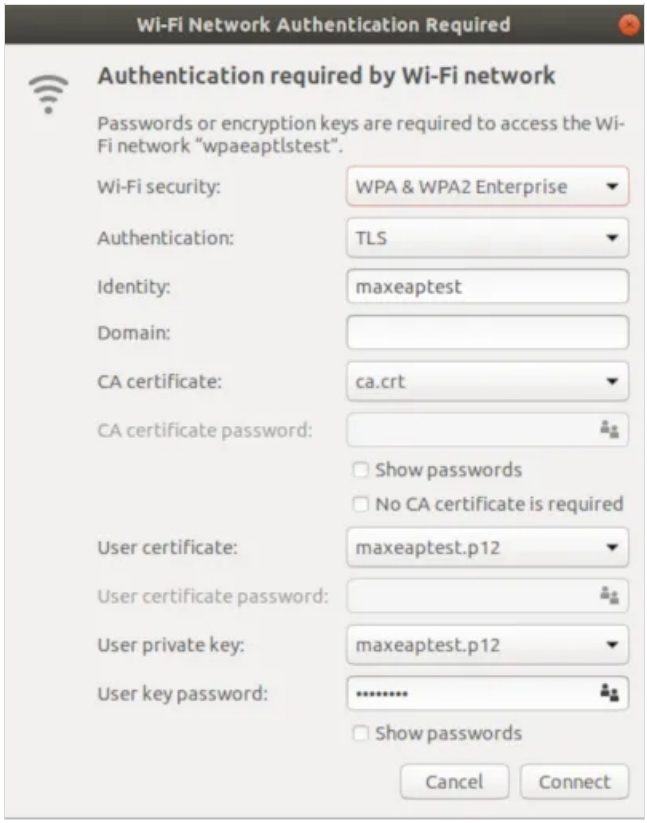
Android has native support of the required protocols, although this might depend on the specific version of Android. First, transfer the downloaded certificate to your phone, and then configure the wireless network.



4.4 Linux

4.4.1 User certificate based authentication

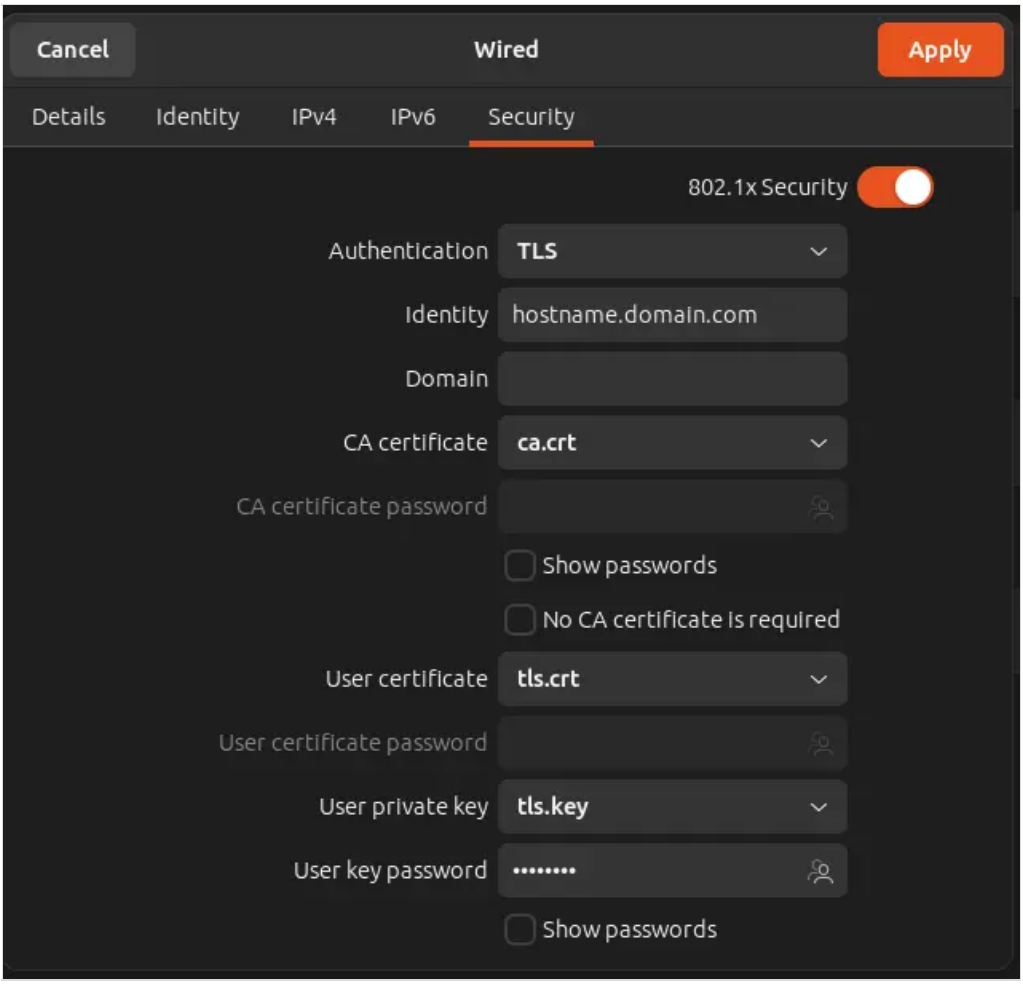
Most Linux clients also have native support and the connection can be configured graphically. Below is a screenshot of Ubuntu 18.04 Network Manager.



4.4.2 Device certificate based authentication

You can copy the p12 bundle previously created on your Linux machine. The CA certificate of WebADM will be needed during the interface setup and can be downloaded at `https://webadm_server_address/cacert`.

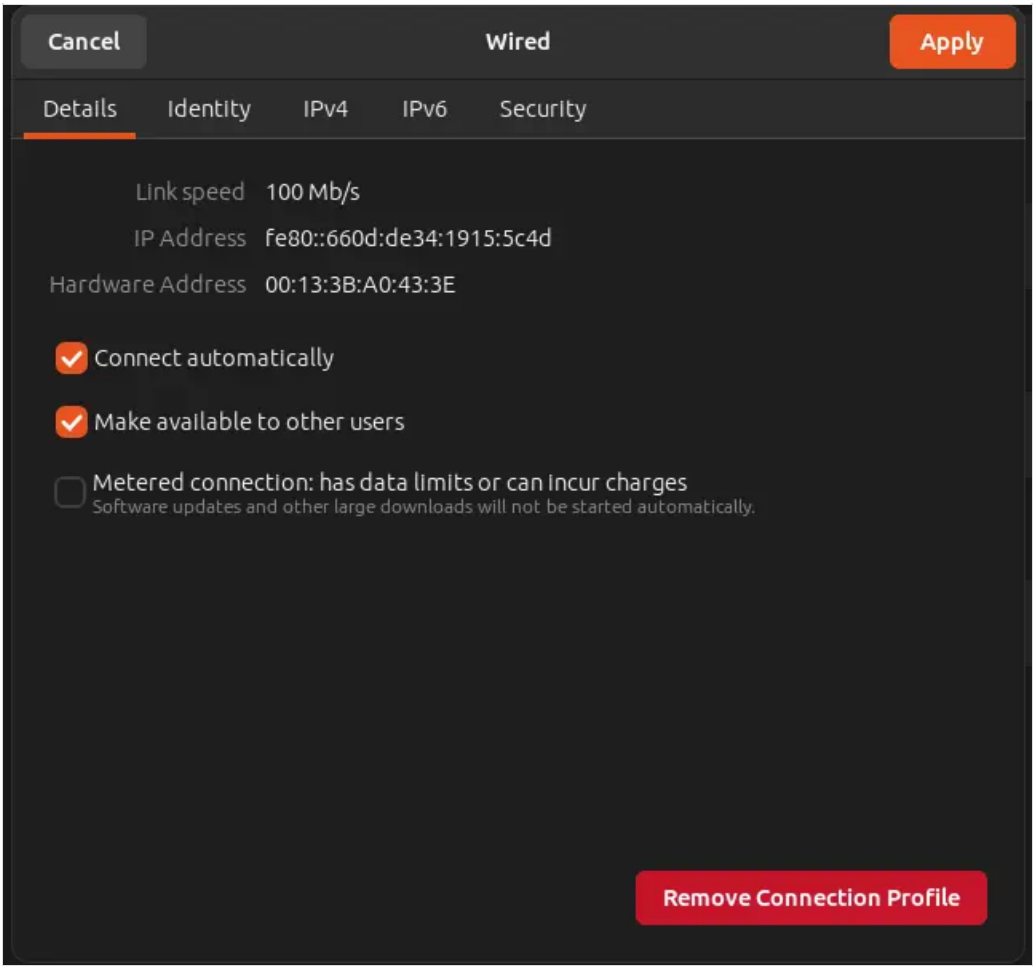
Navigate to the Network configuration of your Linux machine, then edit the Ethernet interface settings.
Configure it like below:



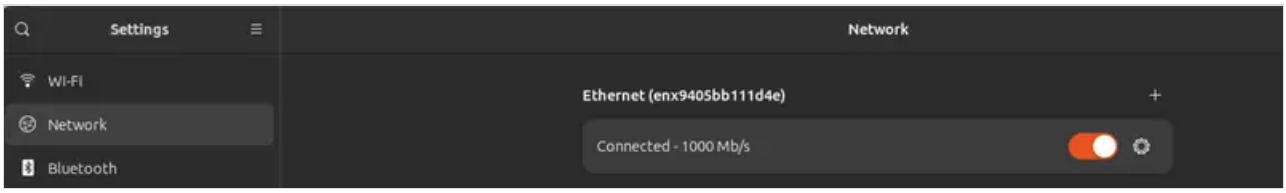
First, enable the `802.1X security` setting. The `Identity` and `Domain` settings are optional as information are retrieved from the client certificate that it is going to be used to establish the authentication and the connection. Configure the CA certificate setting with the CA certificate of your WebADM. Configure the `User certificate` and the `Private key` fields to the certificate p12 bundle previously generated and copied on your Linux machine. Then, provide the p12 password in the `Key password` field.

You can click `Apply` button.

On the `Details` tab, you can enable the setting `Make available to other users` if you want to share the network configuration with other users of that computer:



You should be connected to your network after a successfull EAP-TLS authentication:



5. 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.

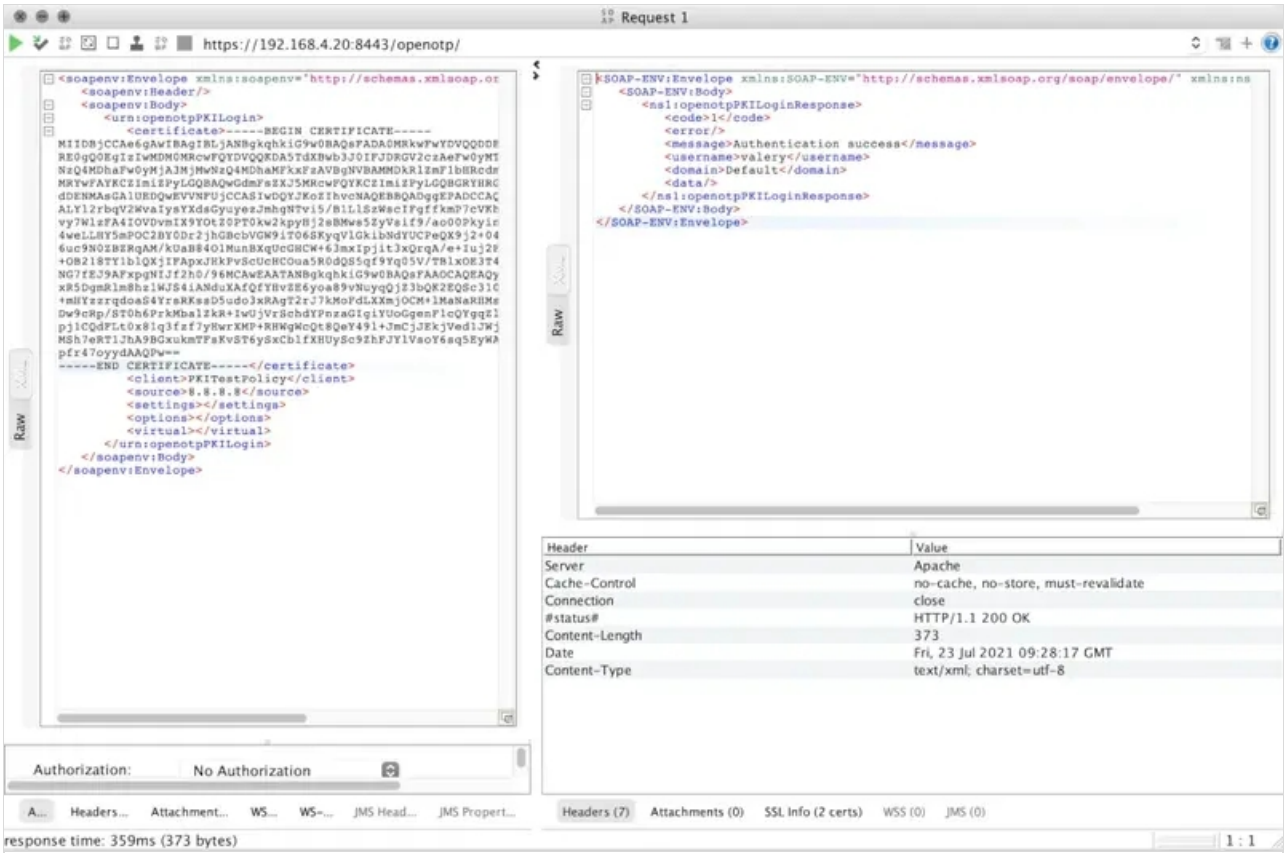
Below, the description of the 2 methods which can be used for this purpose.

```
<!-- 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>
```

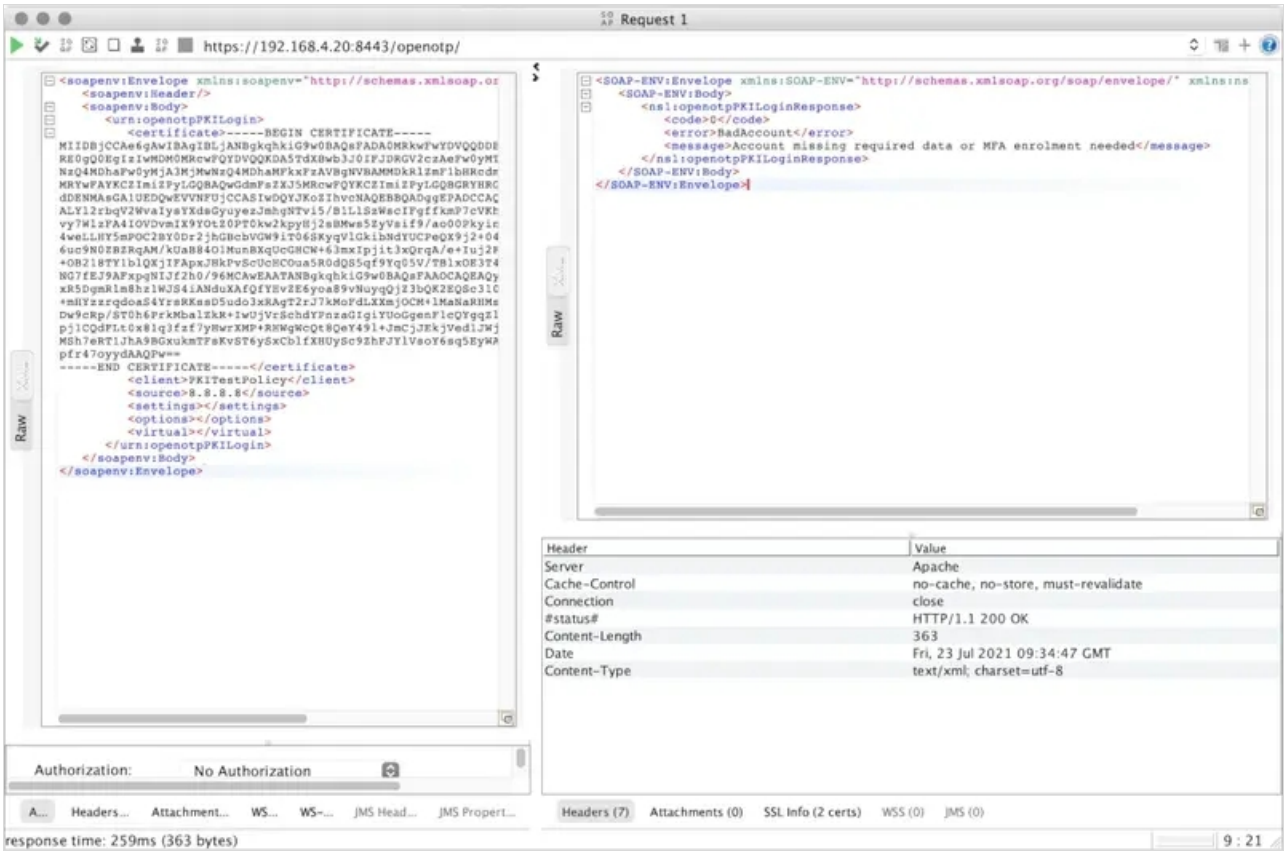
In that documentation, I use the SOAPUI tool to test a certificate-based authentication. What is performed by SOAPUI must be implemented on the client system you want to enable certificate-based authentication. For e.g, if you want to enable certificate-based authentication on your intranet, you must 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.

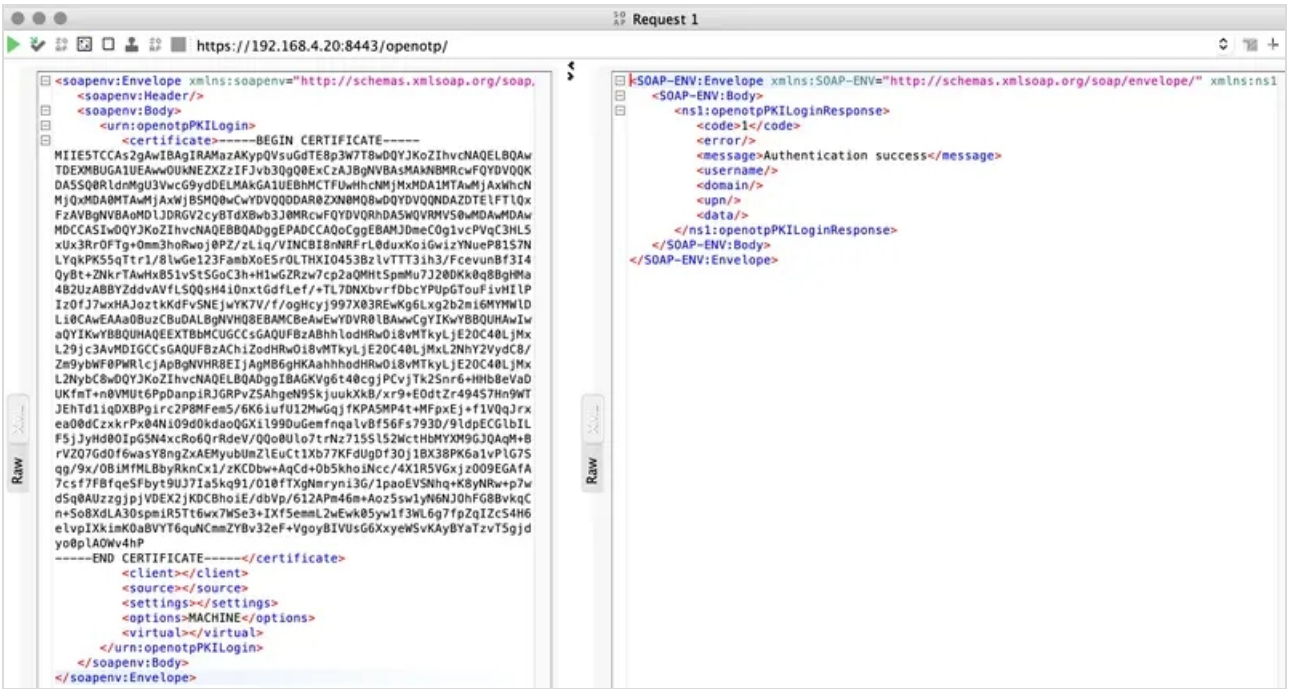
```
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] New openotpPKILogin SOAP request
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] > Certificate: Default\valery (46)
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] > Client ID: PKITestPolicy
[2021-07-23 11:29:29] [10.2.3.6:58127] [OpenOTP:JXU40VZX] > Source IP: 192.168.4.200
[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:O37D471C] New openotpPKILogin SOAP request
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] > Certificate: Default\valery (46)
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] > Client ID: PKITestPolicy
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] > Source IP: 192.168.4.200
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] Registered openotpPKILogin request
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] 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:O37D471C] Resolved source location: US
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] Started transaction lock for user
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] Found user fullname: valery
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] Found 10 user settings: EnableLogin=Yes
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] User has no certificate registered
[2021-07-23 11:34:47] [10.2.3.6:58211] [OpenOTP:O37D471C] Sent failure response
```

For custom implementation of computer certificate based authentication with the OpenOTPPKILogin method, you must pass the `MACHINE` value in the option parameter.

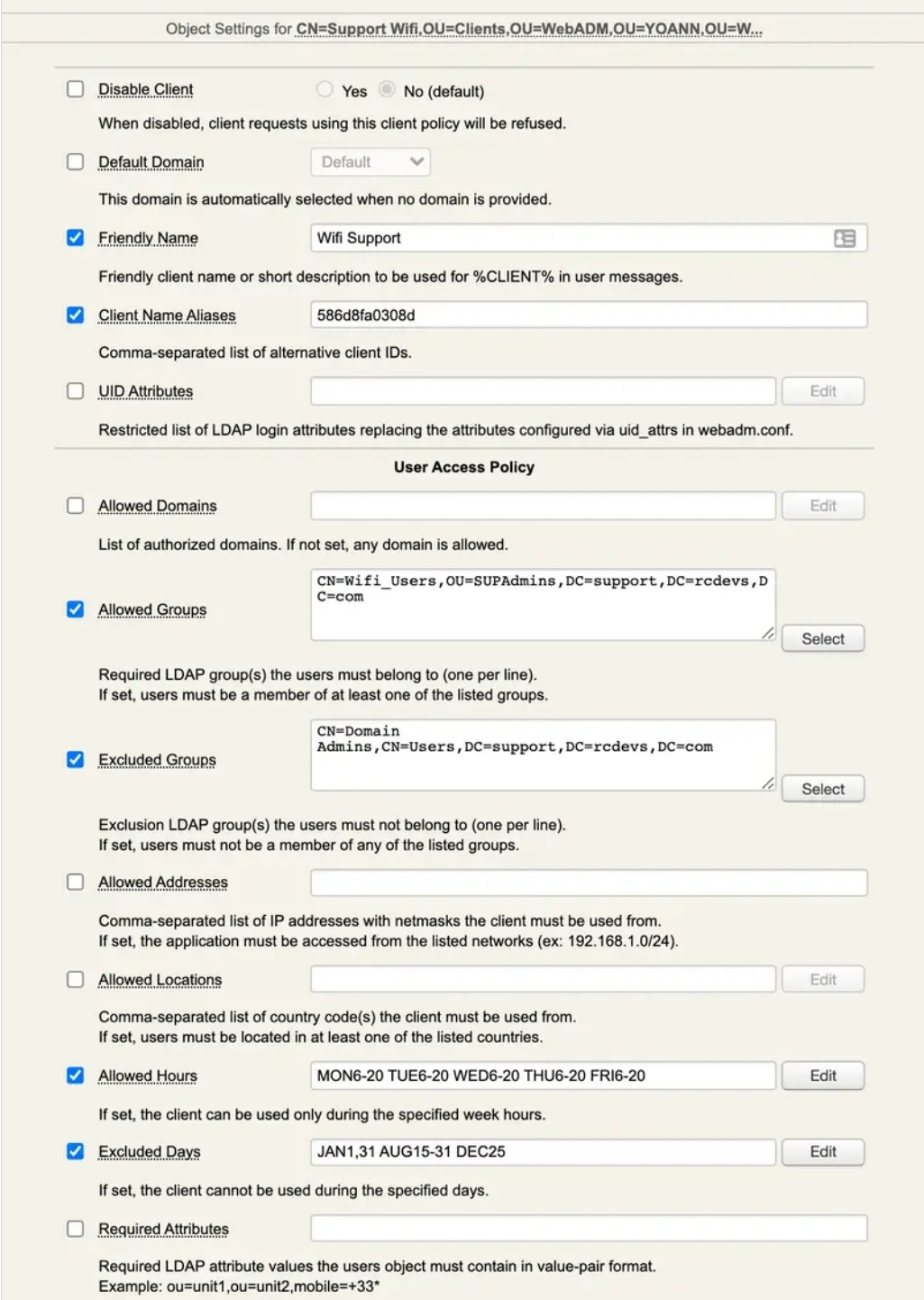


6. Client policies for EAP-TLS, EAP-TTLS logins

6.1 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 Wi-Fi access.

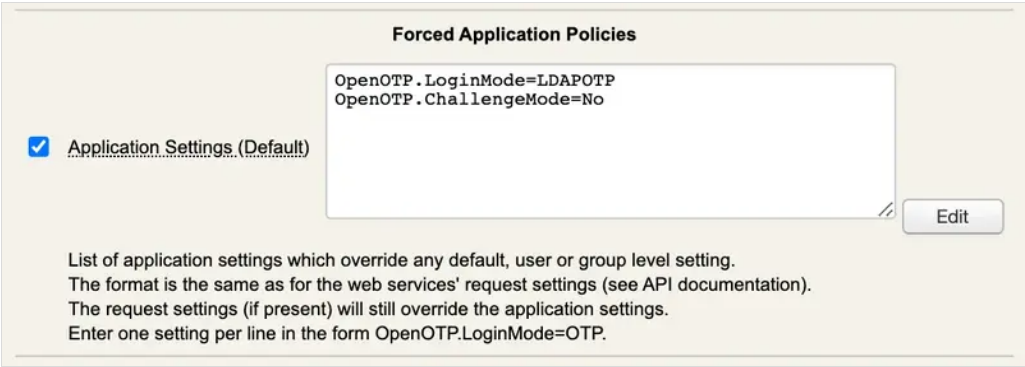
In my `Support Wifi` policy, I allowed my `Wifi_Users` group to log in and disallow my `Domain Admins` group. I've configured allowed hours and days to access that system. I configured the `NAS-Identifier` sent by my EAP system in the `Client Name Aliase`. This is needed to match the WebADM client policy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

6.2 EAP-TTLS policy

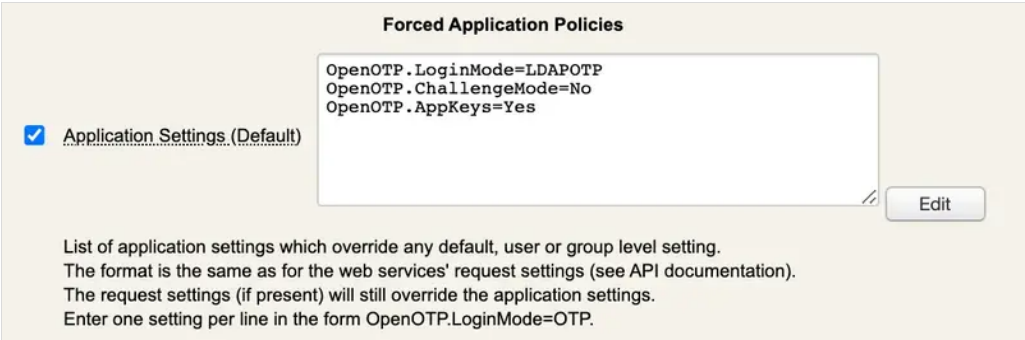
For EAP-TTLS and EAP-GTC client systems, you can create WebADM/OpenOTP client policies. In that scenario, OpenOTP is involved to validate credentials provided by the user during the authentication. In that scenario, you can configure OpenOTP settings. The first OpenOTP setting you need to configure for 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.

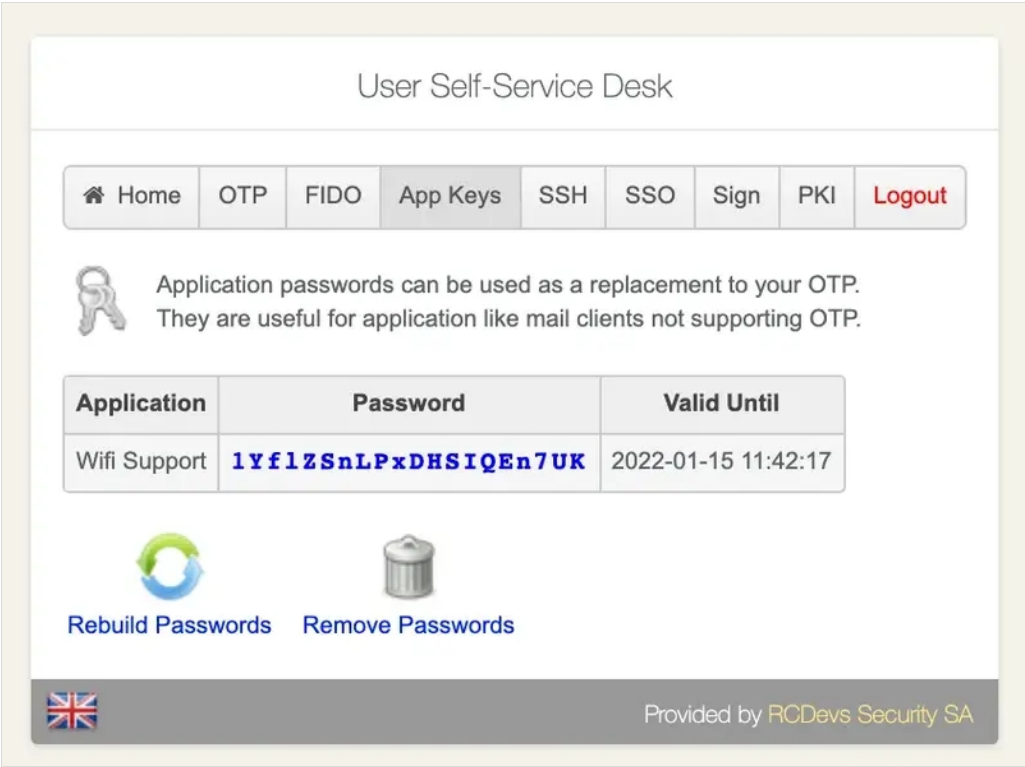
Another interesting feature that can be used here is the implementation of `applications passwords`. When the `applications passwords` feature is enabled for a system, users can use a password randomly generated by WebADM to log in on a specific system. `Application passwords` are generated per client 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.



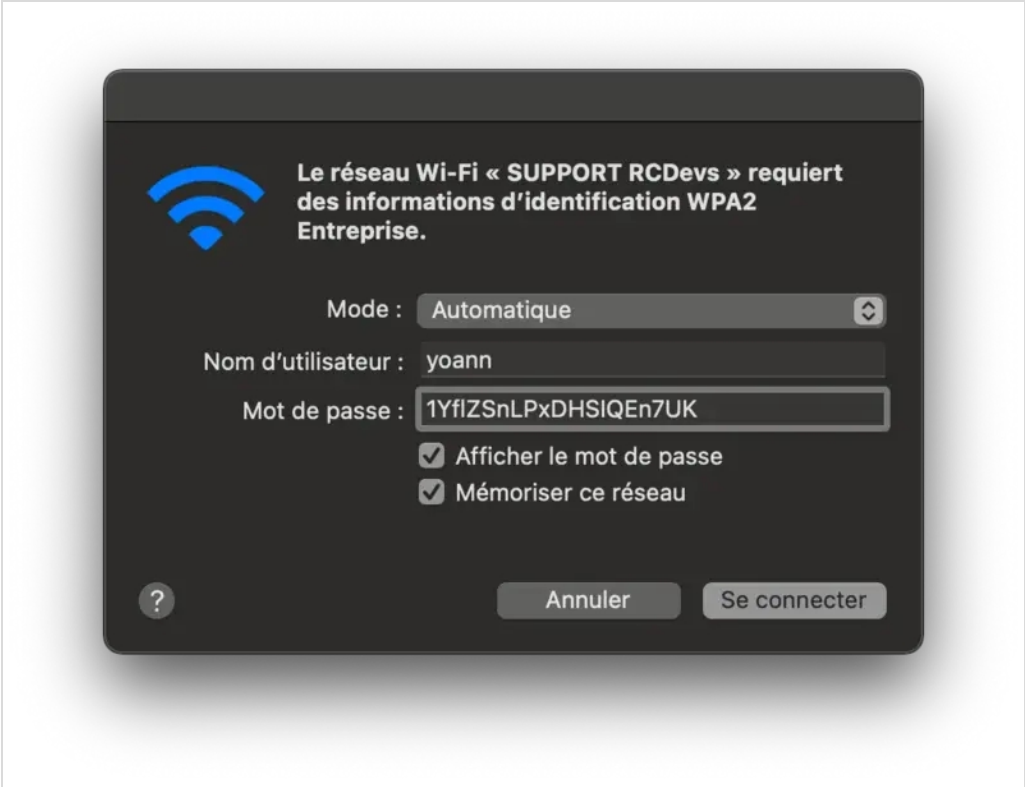
Application password and OpenOTP login mode

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.

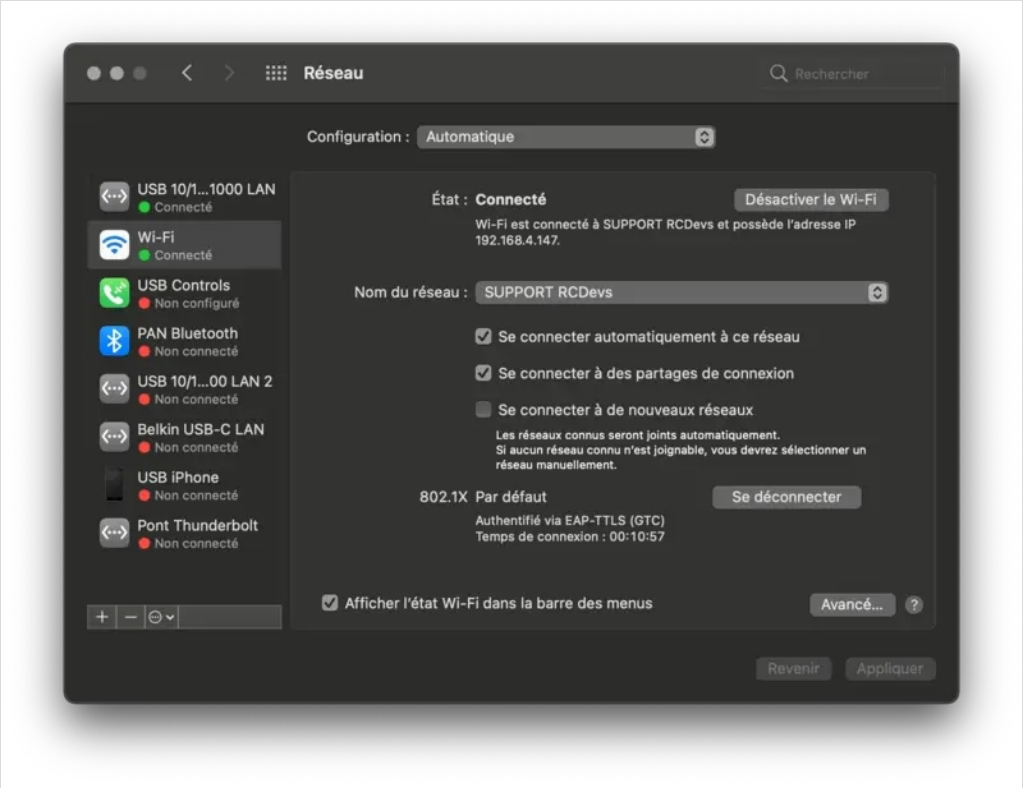
E.g. of `application password` generated for my `Wifi Support` client policy through the User Self-Service Desk.



Usage of my Application password to login :



I am successfully authenticated with my `application password`.



6.2.1 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: foo
[2021-07-19 12:45:12] [192.168.4.20:42248] [OpenOTP:8VTOGU9W] > Password: xxxxxxxxxxxxxxxxxxxx
[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=foo bar,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: foo
[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,OTPLengt
1:HOTP-SHA1-6:QN06-
T1M,DeviceType=FIDO2,U2FPINMode=Discouraged,SMSType=Normal,SMSMode=Ondemand,MailMode=Onc
[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,TokenKey,Tok

[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
```

7. 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.

This manual was prepared with great care. However, RCDevs Security S.A. and the author cannot assume any legal or other liability for possible errors and their consequences. No responsibility is taken for the details contained in this manual. Subject to alternation without notice. RCDevs Security S.A. does not enter into any responsibility in this respect. The hardware and software described in this manual is provided on the basis of a license agreement. This manual is protected by copyright law. RCDevs Security S.A. reserves all rights, especially for translation into foreign languages. No part of this manual may be reproduced in any way (photocopies, microfilm or other methods) or transformed into machine-readable language without the prior written permission of RCDevs Security S.A. The latter especially applies for data processing systems. RCDevs Security S.A. also reserves all communication rights (lectures, radio and television). The hardware and software names mentioned in this manual are most often the registered trademarks of the respective manufacturers and as such are subject to the statutory regulations. Product and brand names are the property of RCDevs Security. © 2024 RCDevs Security S.A., All Rights Reserved