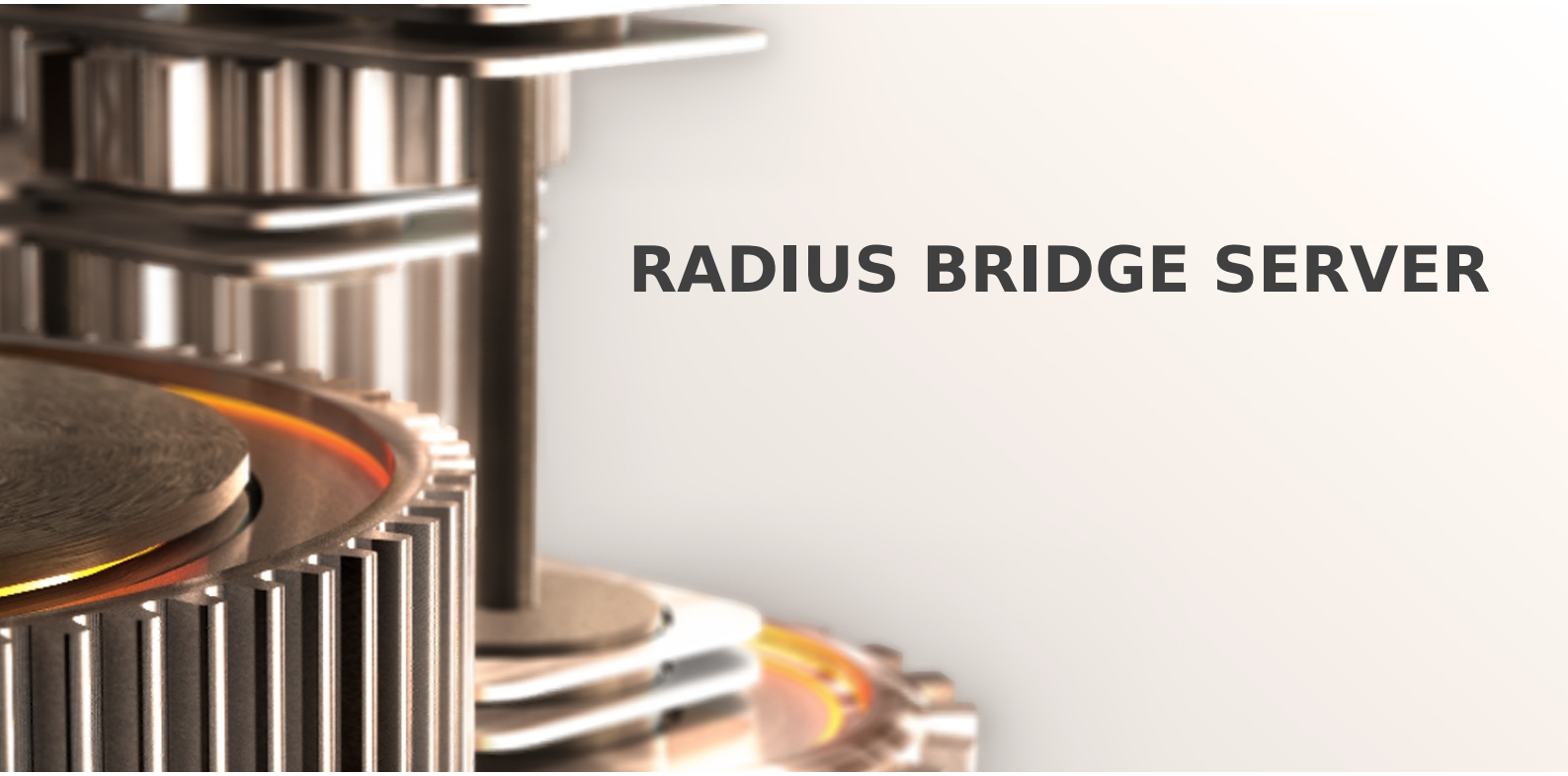

RADIUS BRIDGE SERVER

Radius Bridge Server

Radius

1. Product Documentation

This document is a configuration guide for OpenOTP Radius Bridge (RB). The reader should notice that this document is not a guide for installing and configuring OpenOTP or WebADM. Specific application guides are available through the RCDevs documentation website.

2. Product Overview

OpenOTP Radius Bridge provides the RADIUS RFC-2865 (Remote Authentication Dial-in User Service) API for OpenOTP Authentication Server. Standalone, the OpenOTP server provides SOAP/XML and JSON interfaces over HTTP and HTTPS. By installing and configuring Radius Bridge, you can connect a RADIUS-compliant VPN or any other system supporting the RADIUS authentication protocol.

Radius Bridge is not included in the OpenOTP installation package but in an additional in a self-installer package or through RCDevs repository. It is implemented using the FreeRADIUS software.

FreeRADIUS is the most widely used RADIUS server implementation. More specific FreeRADIUS configurations can be found on the FreeRADIUS web site. The RADIUS RFC-2865 specification provides a Challenge-Response mechanism. OpenOTP challenge authentication mode is also fully supported in the OpenOTP RADIUS API with the RADIUS Challenge-Response. Yet some VPNs do not support RADIUS Challenge-Response. RB also supports concatenated password options for these VPNs. Challenge mode is required for OpenOTP SMS and Mail authentication (in on-demand operating mode).

The Radius Bridge server supports very high loads, is multithreaded and takes advantage of multicore architectures. In clustered environments, it does not require specific RADIUS challenge session tracking as this is completely handled at the OpenOTP level.

3. Product Files and Folders

Find below the RB software installation directory structure and important files.

- `/opt/radiusd/bin/` : Location for RB service binaries and setup.

radiusd: RB executable control script for starting and stopping the server process. To start RB from the command line, issue ./radiusd start. To stop RB, issue ./radiusd stop.

setup: Initial RB setup script automatically run by the self-installer. The setup can be re-run manually at any time.

radtest: Simple RADIUS client test tool. You can use it to check your RB system is working properly without needing to test from the VPN server.

backup: Script to backup your Radius Bridge configuration.

restore: Script to restore your Radius Bridge configuration.

• `/opt/radiusd/doc/` : Location for RB documentation resources.

• `/opt/radiusd/conf/` : Location for RB configuration files.

• radiusd.conf: Main RB configuration file. The setup script should configure this file for you. This file also contains the OpenOTP configurations. This is the most important file and we will see the settings in details in the Configuration section.

• clients.conf: Like in any RADIUS server, you must declare your RADIUS clients (ex. VPN servers). A client consists of the VPN IP address and its RADIUS shared secret. One client must be defined per system connected to RB.

• `/opt/radiusd/lib/` : Location for RB system libraries.

• `/opt/radiusd/lib/dictionaries/` : Location for supported RADIUS vendor dictionaries.

• `/opt/radiusd/libexec/` : Location for RB system executables.

• `/opt/radiusd/logs/` : Location for log files produced by RB.

• `/opt/radiusd/temp/` : Location for temporary files produced by RB.

RB automatically checks the configuration files for syntax errors or mistakes and displays any problem discovered at startup.

4. Installation

4.1 Install with Redhat Repository

On a RedHat, Centos or Fedora system, you can use our repository, which simplifies updates. Add the repository:

```
yum install https://repos.rcdevs.com/redhat/base/rcdevs_release-1.1.1-1.noarch.rpm
```

Clean yum cache and install Radius Bridge:

```
yum clean all
yum install radiusd
```

Radius Bridge is now installed.

4.2 Install with Debian Repository

On a Debian system, you can use our repository, which simplify updates. Add the repository:

```
wget https://repos.rcdevs.com/debian/base/rcdevs-release_1.1.1-1_all.deb
apt-get install ./rcdevs-release_1.1.1-1_all.deb
```

Clean cache and install Radius Bridge:

```
apt-get update
apt-get install radiusd
```

Radius Bridge is now installed.

4.3 Install Using the Self-Installer

The installation of RB is very simple and is performed in less than 5 minutes. Just download the RB self-installer package on RCDevs website and put the installer file on your server. You can use WinSCP to copy the file to your server. To install RB, login to the server with SSH and run the following commands:

```
gunzip radiusd-1.2.x.sh.gz
bash radiusd-1.2.x.sh
```

The installer will install RB in /opt/radiusd/ and will run the setup script automatically. The setup will create the UNIX system user (radiusd), set file and directory permissions, register the startup of RB at system start.

Note

Like other RCDevs software, RB installs its files in one directory only (in /opt/radiusd/). No other is copied to your system but the startup links.

5. Configuration

5.1 Setup Script

A setup script is available to configure Radius Bridge. This script can be launched with `/opt/radiusd/bin/setup` command.

```
[root@webadm2 opt]# /opt/radiusd/bin/setup
Checking system architecture...Ok
Enter the server fully qualified host name (FQDN): webadm2.yorcdevs.com
If WebADM is running on this server then press Enter.
Else enter one of your running WebADM server IP or hostname.
Note: You can use host:port if WebADM uses a custom HTTPS port.
Enter WebADM server IP or hostname: 192.168.3.55
Found two server URLs:
> URL1: https://192.168.3.54:8443/openotp/
> URL2: https://192.168.3.55:8443/openotp/
Retrieving WebADM CA certificate... Ok
The setup needs now to request a signed SSL server certificate.
This request should show up as pending in your WebADM interface and an administrator
must accept it!
Waiting 5 minutes for approbation...
```

At this step, you have to log in on the WebADM Administration GUI to approve the SSL certificate request.

Click on the red button at the end of the home page.

On the next screen, you can see the SSL certificate request is pending:

Click on the Accept button and the Radius Bridge setup will continue.

```
Waiting 5 minutes for approbation... Ok
Updating OpenOTP configuration file... Ok
Setting file permissions... Ok
Do you want OpenOTP RADIUS Bridge to be automatically started at boot (y/n)? y
Adding systemd service... Ok
Do you want to register OpenOTP RADIUS Bridge logrotate script (y/n)? y
Adding logrotate script... Ok
OpenOTP RADIUS Bridge has successfully been setup.
```

5.2 Radiusd Configuration File

In this section, we will review and explain all the available OpenOTP settings in Radius Bridge. By default, your RB should work without modifying any setting.

```
#
# OpenOTP RADIUS Bridge configuration
#

# Server URL(s)
# OpenOTP SOAP service URL(s). This is the only mandatory setting.
# When two servers are used, you can set server_url in the form "url1,url2" or you can
preferably
# comment the server_url line and configure server_url1 and server_url2.
server_url1 = https://192.168.3.54:8443/openotp/
server_url2 = https://192.168.3.55:8443/openotp/

# Request routing policy
# Request routing policy when two server URLs are defined.
# Ordered: First server is preferred (default). When down, second server is used.
# Balanced: Server is chosen randomly. When down, the other is used.
# Consistent: One specific user ID is always routed to the same server (per user
routing).
#server_policy = "Ordered"

# Password mode (deprecated in flavor of WebADM Client Policies!)
# 0: Let OpenOTP automatically handle passwords and concatenation (default).
# 1: RADIUS Access Request transports LDAP password and Access Challenge transports OTP
password.
# 2: RADIUS Access Request transports OTP password (no challenge).
# 3: RADIUS Access Request transports both LDAP and OTP passwords concatenated.
#    The RADIUS password contains the LDAP password followed by the OTP password.
#    Requires either password_separator or otp_length setting below.
# 4: RADIUS Access Request transports both OTP and LDAP passwords concatenated.
#    The RADIUS password contains the OTP password followed by the LDAP password.
#    Requires either password_separator or otp_length setting below.
#password_mode = 0

# OTP length (deprecated)
# With password_mode 3 and 4, radiusd need to know the length of the OTP passwords when
no
# password_separator is set in order to locate the OTP and LDAP parts in the
concatenated
# password value. The otp_length and password_separator settings cannot be used at the
same time.
#otp_length = 6

# Password separator (deprecated)
# With password_mode 3 and 4, radiusd requires a separator character when no otp_length
is set
# in order to locate the OTP and LDAP parts in the concatenated password value.
#password_separator = "+"

# Challenge suffix
# Suffix to be added to the challenge message.
#challenge_suffix = ": "

# Default domain
```

```
# Default domain
# This domain name can be used to override the default domain on the OpenOTP
configuration.
#default_domain = "mydomain"

# Domain separator
# This is the separator character to be used when the domain is provided in the
username.
# For example if '\' is used then username with domain can be in the form
domain\username.
# By default there is no domain sperator.
domain_separator = "\\"

# Support ActiveDirectory UPNs
# When enabled, the user domain is extracted from the UID value when a Active Directory
# User Principal Name (UPN) is provided as usersername (ex. user@domain.com).
# In this example domain.com is provided to OpenOTP as domain name.
#upn_domain = yes

# Client attribute
# This is the RADIUS attribute which contains the client ID to be sent to OpenOTP.
# If this attribute is not found then the NAS IP address is sent as client ID.
# Multiple attributes can be used in the form "NAS-Identifier,NAS-IP-Address".
# By default the NAS-Identifier, NAS-IP-Address and NAS-IPv6-Address attributes are
used.
# If Calling-Station-Id or Called-Station-Id is used here and contains a ':' character,
then only
# the trailing part after the ':' separator is used.
#client_attribute = "NAS-Identifier"

# Source attribute
# This is the RADIUS attribute in which the RADIUS client can pass the end user source
IP address to
# OpenOTP. Attribute must be of type IPAddr.
# By default the source attribute is set to Calling-Station-Id & PaloAlto-Client-
Source-IP.
#source_attribute = "Calling-Station-Id,PaloAlto-Client-Source-IP"

# Context attribute
# This is the RADIUS attribute in which the RADIUS client can pass the end user device
ID address to
# OpenOTP. Attribute must be of type String.
# By default the context attribute is not set (ignored).
#context_attribute = "Calling-Station-Id"

# Settings attribute (deprecated in flavor of WebADM Client Policies!)
# This is the RADIUS attribute in which the RADIUS client can pass user settings to
OpenOTP.
# If the attribute is present in the RADIUS request, it will override any existing user
setting
# from the user_settings setting above. Attribute must be of type String.
# By default the settings attribute is not set (ignored).
#settings_attribute = "Filter-Id"
```

```
#settings_attribute = "Filter-Id"

# User settings
# Fixed list of OpenOTP policy settings to be passed via the OpenOTP API.
#user_settings = "LoginMode=LDAPOTP,OTPType=SMS"

# Client certificate (use if OpenOTP is configure with "Require Client Certificate")
#cert_file = "/opt/radiusd/conf/radiusd.pem"
#cert_password = ""

# Trusted CA (WebADM CA certificate)
# Copy the WebADM CA file in conf/ca.crt and set the ca_file to enforce SSL server
trust.
#ca_file = "/opt/radiusd/conf/ca.crt"

# SOAP timeout
# This is the SOAP request TCP timeout. Set the RADIUS timeout to a lower value on your
RADIUS client.
# If you use OpenOTP Simple-Push login, then you must set the timeout to 30 secs and
you must set the
# RADIUS timeout on your client (NAS) to 30 secs.
#soap_timeout = 30

# Status cache
# When two servers are configured, RadiusBridge can check the server statuses at
regular intervals by
# trying TCP socket connections. The status_cache is the polling interval between 10
and 600 seconds.
# By default, the server statuses are re-checked every 30 seconds. Use 0 disables the
status requests.
#status_cache = 30

# RADIUS reply attributes
# This is a fixed list of attribute and values to be sent back to the RADIUS clients in
Access-Accept
# packets. The syntax is the standard RADIUS value pairs (ie.
attr1=value1,attr2=value2,...).
# Note: The attributes must be present in the local dictionaries (in
lib/dictionaries/).
#reply_attributes = "Juniper-Allow-Commands=\"XXX\",Juniper-Deny-Commands=\"YYY\""

# No success/failure message
# If set to 'yes', then no RADIUS Reply-Message attribute is sent in the Access-Success
and/or
# Access-Failure response. This is useful for some broken RADIUS clients which refuse
the reply
# message attributes in the Access-Request responses.
#no_success_message = no
#no_failure_message = no

# No response delay
# You can configure RB to delay its Access-Reject responses when the OpenOTP server
does not respond.
```

```
does not respond.
# Setting a delay allows RADIUS clients to enforce a failover policy if they do not
receive a RADIUS
# response within a configured timeout. Without the no_response_delay (RB default) the
client gets a
# RADIUS failure response and does also not failover to a secondary server.
#no_response_delay = 15

# MS DirectAccess Probe
# Enable this setting only if you are using Microsoft VPN with DirectAccess server.
# DirectAccess servers check the RADIUS server status via RADIUS probes requests which
are sent to
# OpenOTP via Status requests.
#directaccess_probe = no
#daprobe_username = "DAProbeUser"
#daprobe_password = "DAProbePass"

# Users with OpenOTP transaction lock disabled
# Use ONLY with stress-testing usersw which require concurrent login transactions.
#nolock_usernames = "user1,user2"

# Users for which LDAP credentials will be cached in OpenOTP
# Use ONLY with system polling users generating a lot of OpenOTP LDAP requests.
#cached_usernames = "user1,user2"

# Users to be rejected without sending an OpenOTP request
#denied_usernames = "root"

# FIDO-U2F support
# Enable U2F over RADIUS with RCDevs vendor-specific U2F dictionary (currently
unsupported).
# Uses dictionary attributes from /opt/radiusd/lib/dictionaries/dictionary.rcdevs.
#u2f_support = no

# Short RADIUS timeout fix
# Enable support for RADIUS servers not supporting the 30 seconds' request timeout
required by
# OpenOTP Push Login. You should enable this option if you are using a Cisco ASA VPN
server.
#fix_timeout = no
```

5.2.1 Server Endpoint URL(s) (server_url)

This is the OpenOTP SOAP endpoint URL(s). And this is the only mandatory RB setting. When WebADM and RB are installed on the same server, the server URL should be set to http://127.0.0.1:8080/openotp/. If WebADM and RB are installed on different servers it should use OpenOTP SSL port and be set to `https://<WEBADMSERVER>:8443/openotp/`.

It is possible to configure two different server URLs in the form “url1,url2” (separated by a comma), or alternatively, you can comment the server_url line and configure server_url1 and server_url2. When two servers are configured, you may also choose a request routing policy as explained below.

5.2.2 Request Routing Policy (server_policy)

If two server URLs are defined in server_url, you can optionally configure a request routing policy (ie. the server selection policy). There are three policies available:

- Ordered: The first server is always preferred. When it does not respond, the second server is used.
- Balanced: The server is chosen randomly for each request. When it does not respond, the other is used.
- Consistent: The server selection depends on the user ID. A request for one specific user is also always routed to the same server. If it does not respond, the other server is used.

5.2.3 Status Cache Time (status_cache)

When two servers are configured, RadiusBridge can check the server statuses at regular intervals by sending OpenOTP status requests. The status_cache is the polling interval between 10 and 600 seconds. By default, the server statuses are re-checked every 60 seconds. Use the value '0' disables the OpenOTP status request polling mechanism.

5.2.4 Password Mode (password_mode)

The RADIUS protocol can transport one password at a time. But the OpenOTP API supports passing both LDAP and OTP password in one request (in two different fields). Also, when OpenOTP is used with both LDAP and OTP passwords for the authentication (i.e. LDAPOTP LoginMode in OpenOTP), several mechanisms can be used with RB:

1. The RADIUS Access-Request transports the LDAP password. Then the RB server issues a RADIUS Access-Challenge and a RADIUS Challenge-Response request transport the OTP password. The user is also prompted for his OTP after having entered his LDAP password.
2. The RADIUS Access-Request transports a concatenated form of the LDAP and OTP passwords in the same RADIUS Access-Request. Multiple concatenation options are available.

Alternatively, RB can work with OpenOTP LDAP-only (i.e. LDAP LoginMode in OpenOTP) and OTP-only (i.e. OTP LoginMode in OpenOTP). In that case, the RB is able to transport only the LDAP or OTP password in the RADIUS Access-Request.

> ⚠ Note
>
> Current versions of the OpenOTP server are able to handle password concatenation at the OpenOTP server level. For this, you only need to configure a Client Policy for your RADIUS client(s) in WebADM and set the Challenge Support to No in the Application Settings. You should keep the password_mode to its default value (0) for automatic password decoding. Please look at section 6 for more information about WebADM Client Policies.

The password modes supported by RB are as follows :

- password_mode = 0: This is the default operating mode where Radius Bridge lets OpenOTP handle the request passwords automatically. This mode uses the OpenOTP v1.1 SimpleLogin API method. This mode is highly recommended for common

integrations.

- password_mode = 1: The RADIUS Access-Request transports LDAP password and Access Challenge transports OTP password. This is the default if the setting is not specified. This mode works with OpenOTP LDAP only and LDAPOTP with challenge.
- password_mode = 2: The RADIUS Access-Request transports only the OTP password (no challenge).
- password_mode = 3: The RADIUS Access-Request transports both LDAP and OTP passwords concatenated. The RADIUS password contains the LDAP password followed by the OTP password. This setting requires either password_separator or otp_length setting below.
- password_mode = 4: The RADIUS Access-Request transports both the OTP and LDAP passwords concatenated. The RADIUS password contains the OTP password followed by the LDAP password. Requires either password_separator or otp_length setting below.
- password_mode = 5: The RADIUS Access-Request transports both user ID and OTP password concatenated. The RADIUS username contains the user ID followed by the OTP password. Requires either password_separator or otp_length setting below.

Here is a summary of the possible password policies in RADIUS Bridge:

OpenOTP LDAP password only (LDAP LoginMode):

- Use password_mode = 0 (default) or 1
- Users provide the LDAP password in the Radius Access-Request.

OpenOTP OTP password only (OTP LoginMode):

- Use password_mode = 0 (default) or 2
- Users provide the OTP password in the Radius Access-Request.

OpenOTP LDAP+OTP passwords (LDAPOTP LoginMode):

Option 1) With Challenge mode:

- Use password_mode = 0 (default) or 1
- Users provide the LDAP password in the Radius Access-Request.
- Users are prompted for an OTP via a Radius Challenge-Response.
- Users provide the OTP password in the Radius Access-Challenge.

Option 2) With concatenated mode (LDAPOTP):

- Use password_mode = 3
- Users provide the LDAP password followed by the OTP password in the Radius Access-Request.

Option 3) With concatenated mode (OTPLDAP):

- Use password_mode = 4
- Users provide the OTP password followed by the LDAP password in the Radius Access-Request.

Please look at Appendix A for a more detailed explanation of password modes.

> **Note**
>
> The OpenOTP LoginMode and the RB password mode must be consistent. The OpenOTP LoginMode can be set in the OpenOTP configuration in WebADM and it can be adjusted per LDAP users or groups. And you can create a WebADM Web Service Client object where you can force a login mode for a specific RADIUS client. For example, you can use password_mode 2 (OTP only) and create a client policy object in WebADM for your VPN where you set OpenOTP.LoginMode=OTP (in the Priority Settings of the client object).

It is recommended to let the default configuration with password_mode 0 for common usage. Password modes 1, 2, 3 and 4 should be used only when necessary. Please look at section 6 for details about client policies. Another solution to force the OpenOTP login mode on the OpenOTP server is to use the user_settings RB setting explained below.

5.2.5 OTP Length (otp_length)

With password mode 3 and 4, RB need to know the length of the OTP passwords when no password_separator setting is set, in order to locate the OTP and LDAP parts in the concatenated password value.

> **Note**
>
> The otp_length and password_separator settings cannot be used at the same time. Password separator is highly preferred as your users may be configured with different OTP password lengths.

This setting is deprecated. OTP length for password de-concatenation should be handled by the OpenOTP server when Challenge Support is disabled.

5.2.6 Password Separator (password_separator)

With password_mode 3 and 4, RB requires a separator character is needed when no otp_length setting is set in order to locate the OTP and LDAP parts in the concatenated password value. The default password separator is the '+' character. This setting is deprecated. OTP de-concatenation is handled by the OpenOTP server and a password separator is not needed anymore.

5.2.7 Challenge Suffix (challenge_suffix)

This is a suffix string to be appended to the challenge messages returned by OpenOTP. In some cases, it can be useful to add ': ' for example to the user prompt, for a better display.

5.2.8 Domain Separator (domain_separator)

A RADIUS Access-Request does not provide a standard domain attribute. Yet, WebADM Domain names can be passed in the Radius request as part of the Radius username attribute, by using a Windows NT -like notation (i.e. domain\username or

username@domain). It is also possible to configure what separator character is used when the domain name is provided in the RADIUS username with the domain_separator setting.

By default, no separator is used and RB expects only a username and no domain. When the character '@' is used as domain separator, the domain part is expected to be on the right side of the string in the form username@domain. With any other separator, the domain is expected to be on the left side in the form domain\username.

If you want to use the character "\" as a separator to provide the credential in the form domain\username, then you must configure RB with domain_separator = "\".

5.2.9 UPN Domain Support (upn_domain)

Set this setting to Yes if you use ActiveDirectory LDAP with User Principal Names (UPN). UPNs are globally unique login names like email addresses (ex. user@company.com). The UPN contains the DNS domain as part of the user ID (after the '@' character). With UPNs, OpenOTP will select the right WebADM Domain based on the UPN domain information. When enabled, RB will pass the UPN domain suffix (ie. right side of the '@') as a domain to the OpenOTP API and the whole UPN as username. For example, if the UPN is user@company.com, then RB will send user@company.com as username and company.com as a domain to OpenOTP.

Note

In WebADM Domains, you can configure the UPN suffixe(s) as Domain Alias in the domain settings.

Active Directory provide two form of UPNs

- Explicit UPN (eUPN): This is the value of the user object's userPrincipalName attribute.
- Implicit UPN (iUPN): This is constructed by concatenating the value of the user object's samAccountName attribute with the value of the AD domain's FQDN.

The upn_domain setting is designed for using Explicit UPNs. If you need Implicit UPNs, then do not enable upn_domain and just set '@' as domain separator.

5.2.10 Default Domain (default_domain)

It is possible to configure a default_domain in RB to allow users not to provide a domain name if they are part of your default domain.

Note

OpenOTP is configured with a default domain in WebADM. Use this setting only if you want to use a default domain different than the OpenOTP default domain.

> **Note**
>
> You can configure the default domain for a specific VPN client using a WebADM Client Policy object. This is useful when you need a different default domain depending on the RADIUS client. For example, you have two VPNs allowing access to users from two different domains.

5.2.11 User Settings (user_settings)

With the user_settings, you can pass a fixed list of policy settings to OpenOTP in every request. These settings will have a higher priority than any setting defined on the users, groups, client policies and OpenOTP configuration.

Only the public OpenOTP settings can be passed in the OpenOTP requests. For example, you can set user_settings = “LoginMode=OTP,OTPType=TOKEN”. To know the settings names and if they are public, just go to the OpenOTP configuration in WebADM and put the mouse over one setting name. WebADM will display the real setting name (as to be used in the RB user_settings) and its scope (public, private, etc…).

> **Note**
>
> This user setting can be set in a WebADM Web Service Client object too. This is the preferred option as you can configure it from the WebADM interface.

5.2.12 User Settings Attribute (settings_attribute)

It might happen that you want your VPN server to provide a list of OpenOTP user settings as part of the authentication requests. This is also the RADIUS attribute in which the RADIUS client can pass OpenOTP user settings. If the attribute is present in the RADIUS request, it will override any existing user_settings value. By default, no attribute is configured. You can safely use Filter-Id attribute to transport the user settings.

5.2.13 Data Attribute (data_attribute)

This configuration does not exist anymore in Radius Bridge v1.2.4! The documentation is kept for older versions of Radius Bridge.

You might need to return a specific attribute to the RADIUS client. For example, you want to return a user role to a Juniper SSL-VPN. In WebADM, you can set a Reply Data in the OpenOTP user settings. This is also the RADIUS attribute in which RB will return the content of the OpenOTP Reply Data found in the LDAP user.

For example, the user has a Reply Dataset to MyRole. Then the VPN server will receive a RADIUS attribute Filter-Id=”MyRole”.

> **Note**
>
> This setting is ignored if the data_is_vps setting is set to ‘yes’.

> **Note**
>
> The attributes must be present in the local dictionaries (in lib/dictionaries/).

5.2.14 Data Separator (data_separator)

This configuration does not exist anymore in Radius Bridge v1.2.4! The documentation is kept for older versions of Radius Bridge.

You can return several instances of the data attribute by specifying a separator character and set a list of Reply Data in the LDAP users, separated with the separator character. RB will create one data attributes per Reply Data in the RADIUS response. If no separator is specified, the Reply Data is copied to one unique data_attribute.

For example, the user has a Reply Dataset to MyRole1, MyRole2. Then the VPN server will receive two RADIUS attributes: Filter-Id=”MyRole1” and Filter-Id=”MyRole2”.

> **Note**
>
> This setting is ignored if the data_is_vps setting below is set to ‘yes’.

> **Note**
>
> The attributes must be present in the local dictionaries (in lib/dictionaries/).

5.2.15 Data with Value-pair (data_is_vps)

This configuration does not exist anymore in Radius Bridge v1.2.4! The documentation is keep for older versions of Radius Bridge. OpenOTP includes a new policy setting called RADIUS Attributes which is used to configure per user or group RADIUS reply attributes.

If this setting is set to ‘yes’, then RB assumes the user Reply Data contain a list of RADIUS attribute-value pairs. In that case, the RADIUS attributes defined in the Reply Data are created by RB with their values and returned to the RADIUS client.

For example, the user has a Reply Dataset to Juniper-Allow-Commands=“CMD1”, Juniper-DenyCommands=“CMD2”. Then the VPN server will receive two RADIUS attributes: Juniper-AllowCommands=“CMD1” and Juniper-Deny-Commands=“CMD2”.

RB supports per RADIUS client value-pair filtering. For example, you might want to set different roles for a user depending on the VPN. In this case, let’s say you want to use Filter-Id as role attribute on both VPNs but the user a Role1 on VPN1 and Role2 on VPN2. Then you just set VPN1:Filter-Id=”Role1”, VPN2:Filter-Id=“Role2” in the user Reply Data. The VPN1 server will receive the RADIUS attributes Filter-Id=”Role1” and the VPN2 server will receive VPN1.FilterId=”Role2”. You can use either character ‘:’ or ‘.’ as client filter separator.

> **Note**
>
> VPN1 and VPN2 in our example correspond to the NAS-Identifier passed by the RADIUS client, or the IP address if NAS-Identified is not provided.

> **Note**
>
> The attributes must be present in the local dictionaries (in lib/dictionaries/).

5.2.16 RADIUS Reply Attributes (reply_attributes)

This is a fixed list of static RADIUS attribute value-pairs to be always sent back to the RADIUS clients in the Access-Accept responses. The values will be combined with the RADIUS reply attributes which are configured in WeBADM. The syntax is the standard RADIUS value pairs (ie. attr1=value1,attr2=value2,…). Example : reply_vps = “Juniper-Allow Commands="XXX",JuniperDeny-Commands="YYY"”.

> **Note**
>
> The attributes must be present in the local dictionaries (in lib/dictionaries/).

5.2.17 Client ID Attribute (client_attribute)

RADIUS Bridge uses this attribute to send the client ID to OpenOTP. This attribute is set to NASIdentifier, NAS-IP-Address and NAS-IPv6-Address by default (attributes are tried in order). When none of the configured is found, the requestor IP address is sent as client ID.

5.2.18 Source IP Attribute (source_attribute)

The RADIUS client can optionally forward the IP address of the end-user to RadiusBridge. This IP address is used by the OpenOTP audit and with WebADM location policies defined on Domain and Client Policy configuration objects. The user RADIUS source IP attribute is not provided by most VPN vendors. Yet, with some VPNs like Cisco ASA use the Calling-Station-Id to provide the user address. By default, the source attribute is set to Calling-Station-Id.

The attribute must be defined in the RADIUS dictionary and be of type IPAddr or String. The attribute value will be ignored if it does not contain a valid IP address.

5.2.19 Context ID Attribute (context_attribute)

The RADIUS client can optionally forward the device ID (ie. the MAC address of the user’s connecting device) to RadiusBridge. The device ID is used by the OpenOTP contextual authentication feature. By default the context attribute is not configured (ignored). With Cisco Wifi, you may set it to Calling-Station-Id which may provide the MAC address of the client Wifi device. The attribute must be defined in the RADIUS dictionary and be of type String.

5.2.20 SOAP Timeout (soap_timeout)

This is the OpenOTP SOAP requests' timeout. It should be equal or lower than the RADIUS timeout configured on your RADIUS client(s). The minimal authorized timeout is 5 seconds. The default timeout is 30 seconds. If you use the OpenOTP Simple-Push Login method then the timeout value should be set to 30 seconds. If you don't use Simple-Push then the timeout value should be set to 10 seconds.

5.2.21 CA Certificate file (ca_file)

You can copy your WebADM CA's public certificate file, in the Radius Bridge configuration folder and point the ca_file to the certificate file, if you need to enforce OpenOTP server authentication with SSL. This configuration works only if OpenOTP is a remote service accessible via SSL.

5.2.22 No Success/Failure Messages (no_success_message & no_failure_message)

You can prevent RADIUS Reply-Message attributes to be sent in the Access-Success and/or Access-Failure response. This is useful for some broken RADIUS clients which refuse the reply message attributes in the Access-Request responses. It should not be used otherwise.

5.2.23 No OTP Response Delay (no_response_delay)

You can configure RB to delay its Access-Reject responses when the OpenOTP server does not respond. Setting a delay allows the RADIUS clients to enforce a failover policy if they do not receive a RADIUS response within a configured timeout. Without the no_response_delay (RB default) the client gets a RADIUS failure response and does also not failover to a secondary server. Use this feature only if you configure your RADIUS client(s) with several RADIUS servers.

5.2.24 MS DirectAccess Probe (directaccess_probe & daprobe_username & daprobe_password)

Enable DirectAccess probe ONLY if you are using Microsoft VPN with DirectAccess server! DirectAccess servers check the RADIUS server status via RADIUS probes requests which are sent to OpenOTP in Status requests. Successful DirectAccess probes return access-success responses to the Microsoft VPN server. You must also use this setting with extreme caution. You must NEVER enable probe requests for any other RADIUS client.

5.2.25 nolock_usernames & cached_usernames & denied_usernames

Nolock_usernames is the list of usernames for which you want OpenOTP to disable transaction locks. Cached_usernames is the list of usernames for which you want OpenOTP to cache the login results for a short amount of time. Denied_usernames is the list of usernames which are immediately denied.

5.2.26 U2F Support (u2f_support)

Enable U2F over RADIUS with RCDevs vendor-specific U2F dictionary (currently supported by Viscosity VPN client). Uses dictionary attributes from `/opt/radiusd/lib/dictionaries/dictionary.rcdevs`.

5.3 RADIUS Clients Configuration File

Your RADIUS clients (ex. VPN server) must be registered in the `/opt/radiusd/conf/clients.conf` file to be able to communicate with the RB server. A client configuration looks this:

```
client my_vpn {
    ipaddr = 192.168.0.10
    secret = testing123
}
```

You need to set the IP address of your RADIUS client and the shared RADIUS secret. On the VPN side, you will configure a RADIUS server with its IP address (ie. the RB server IP address), and you will set the same secret.

Note

Always prefer setting no RADIUS retries (retries=0) on the RADIUS configuration of your VPN when you use OpenOTP challenge mode.

6. Radius Bridge and WebADM Client Policies

As we have seen for password modes, it can be useful to create WebADM client policies for your RADIUS clients. For example, you might need to set a default domain for a specific VPN or you want to restrict access to users who are members of a specific LDAP group. You may also need to define different access policies for your VPNs. For example, you want all users to use OTP Login Mode for a VPN, whatever Login Mode is configured for the user or in OpenOTP. For all these reasons, you can create WebADM Web Service Client objects. A WebADM Web Service Client object can be defined if you need to assign an access control policy for a specific client application (which uses the SOAP or RADIUS APIs), or if you want to force some WebADM application settings for a client application. You can also define per-client application profiles in WebADM using Client objects.

By defining a Web Service Client, you can, for example, restrict access to the Client application for some LDAP authorized groups or prevent some groups to use the application. You can even restrict the application to work with some specific WebADM Domains.

Another feature of the Client is that you can define some Web Application settings which will always be enforced for the client application whatever setting is set in the users or its groups. For example, you want one VPN to authenticate users through OpenOTP with OTP only passwords and Token whatever policy is defined for the user, and you want your internal systems to authenticate users with LDAP only.

To create a Client object, you must know your client application ID. The WebADM Client object must have the same name as the client ID. The ID is typically the Client name that appears in the WebADM Log Viewer for Web Services. The Client ID is generally provided in the client requests in the *client* SOAP attribute. With RADIUS, it is the NAS-Identifier. If this information is not provided by the client, WebADM will use the RADIUS client IP Address as a client name.

6.1 Concatenated Password with Client Policies

When a WebADM Client policy object is configured with Challenge Support disabled and the user policy is set to LDAPOTP, then OpenOTP assumes the passwords are provided in the concatenated form. OpenOTP will also de-concatenates the LDAP and OTP passwords which must be provided with the LDAP password followed by the OTP password (without any separator character). This new feature can replace the RadiusBridge password_mode configuration for concatenated passwords. You can simply let RB with its default password_mode (mode 0) and let OpenOTP do the job.

7. PPTP/L2TP VPNs

RB supports PPTP and L2TP VPN servers with PAP authentication only. CHAP and similar password protocols using hashed values are not supported. If you use PPTP with VPN clients such as Windows integrated VPN client, be sure to configure the VPN client and server with PAP authentication.

> **Note**
>
> PAP uses a cleartext password transport is not recommend if you use OpenOTP with LDAP passwords. In this case, you should consider L2TP (PPP over IPSec) with PAP and not PPTP with PAP. With L2TP the VPN is established inside an IPSec channel and the PAP password is secured. A PPTP VPN with PAP remains acceptable if you use OpenOTP with OTP only since an OTP password is one-time and cannot be replayed.

> **Note**
>
> The PAP concern is relevant only with PPTP VPN. Common VPN vendor like Cisco, Juniper, Checkpoint, F5, etc… do not rely on the PAP protocol.

8. Troubleshooting

Radius authentication can fail for many reasons, the following chapters describe how to troubleshoot it.

8.1 Radiusd Status Check

Check if the OpenOTP RADIUS Bridge service is running:

```
[root@rcvm8 ~]# /opt/radiusd/bin/radiusd status
OpenOTP RADIUS Bridge is running with PID 1635.
[root@rcvm8 ~]# systemctl status radiusd
● radiusd.service - OpenOTP Radius Bridge
   Loaded: loaded (/usr/lib/systemd/system/radiusd.service; enabled; vendor preset:
disabled)

>>>> NOTE
   Active: active (running) since Tue 2020-03-31 18:42:35 CEST; 2 days ago
NOTE <<<<

  Process: 1024 ExecStart=/opt/radiusd/bin/radiusd start (code=exited,
status=0/SUCCESS)
 Main PID: 1635 (rcdevs-radiusd)
    Tasks: 6 (limit: 11501)
   Memory: 85.9M
   CGroup: /system.slice/radiusd.service
           └─1635 rcdevs-radiusd
```

Check if the OpenOTP RADIUS Bridge service `rcdevs-radius` is listening on right IP Address and Port. May need to install the package `net-tools` to run the command `netstat`.

```
[root@rcvm8 ~]# yum install net-tools
...
Installed:
  net-tools-2.0-0.51.20160912git.el8.x86_64

Complete!
[root@rcvm8 ~]# netstat -tulpn
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 0.0.0.0:5355            0.0.0.0:*               LISTEN
1657/systemd-resolv
tcp        0      0 0.0.0.0:10636           0.0.0.0:*               LISTEN
1208/rcdevs-ldproxy
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN
2032/webadm-httpd
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
2032/webadm-httpd

>>>> NOTE
tcp        0      0 0.0.0.0:1812            0.0.0.0:*               LISTEN
24467/rcdevs-radius
NOTE <<<<

tcp        0      0 0.0.0.0:10389           0.0.0.0:*               LISTEN
1208/rcdevs-ldproxy
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
1023/sshd
tcp        0      0 0.0.0.0:8443            0.0.0.0:*               LISTEN
2032/webadm-httpd
tcp        0      0 0.0.0.0:443             0.0.0.0:*               LISTEN
2032/webadm-httpd
tcp        0      0 0.0.0.0:636             0.0.0.0:*               LISTEN
1284/rcdevs-slapd
tcp        0      0 0.0.0.0:4000            0.0.0.0:*               LISTEN
1994/webadm-session
tcp        0      0 0.0.0.0:389             0.0.0.0:*               LISTEN
1284/rcdevs-slapd
tcp        0      0 0.0.0.0:5000            0.0.0.0:*               LISTEN
1992/webadm-rsignd
tcp6       0      0 :::3306                 :::*                    LISTEN
1138/mysqld
tcp6       0      0 :::5355                 :::*                    LISTEN
1657/systemd-resolv
tcp6       0      0 :::22                   :::*                    LISTEN
1023/sshd
tcp6       0      0 :::4000                 :::*                    LISTEN
```

```
tcp6       0      0 :::4000                 :::*                    LISTEN
1994/webadm-session
udp        0      0 0.0.0.0:18120           0.0.0.0:*
24467/rcdevs-radius
udp        0      0 0.0.0.0:1812            0.0.0.0:*
24467/rcdevs-radius
udp        0      0 0.0.0.0:1813            0.0.0.0:*
24467/rcdevs-radius
udp        0      0 127.0.0.53:53           0.0.0.0:*
1657/systemd-resolv
udp        0      0 0.0.0.0:5355            0.0.0.0:*
1657/systemd-resolv
udp        0      0 127.0.0.1:323           0.0.0.0:*
957/chronyd
udp        0      0 0.0.0.0:1645            0.0.0.0:*
24467/rcdevs-radius
udp        0      0 0.0.0.0:1646            0.0.0.0:*
24467/rcdevs-radius
udp6       0      0 :::5355                 :::*
1657/systemd-resolv
udp6       0      0 ::1:323                 :::*
957/chronyd
[root@rcvm8 ~]#
```

8.2 Connectivity Check

Please check if the ports are not blocked by a firewall with the command `telnet <webadm server> <port number>` and verify radiusd authentication with `tcpdump -i any port 1812`.

```
[root@centos8-client ~]# telnet 192.168.3.232 1812
Trying 192.168.3.232...

>>>> NOTE
Connected to 192.168.3.232.
Escape character is '^]'.
NOTE <<<<

Connection closed by foreign host.
```

```
[root@rcvm8 ~]# tcpdump -i any port 1812 -vv
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144
bytes
13:05:22.645509 IP (tos 0x10, ttl 64, id 3618, offset 0, flags [DF], proto TCP (6),
length 60)
    192.168.3.214.60210 > rcvm8.rcdevs.local.radius: Flags [S], cksum 0x5db7 (correct),
seq 1393734890, win 29200, options [mss 1460,sackOK,TS val 2181395168 ecr 0,nop,wscale
7], length 0
13:05:22.645988 IP (tos 0x0, ttl 64, id 44767, offset 0, flags [none], proto TCP (6),
length 60)
    rcvm8.rcdevs.local.radius > 192.168.3.214.60210: Flags [S.], cksum 0x893d
(incorrect -> 0x4f50), seq 3386925890, ack 1393734891, win 28960, options [mss
1460,sackOK,TS val 4022334050 ecr 2181395168,nop,wscale 7], length 0
13:05:22.646205 IP (tos 0x10, ttl 64, id 3619, offset 0, flags [DF], proto TCP (6),
length 52)
    192.168.3.214.60210 > rcvm8.rcdevs.local.radius: Flags [.], cksum 0xee56 (correct),
seq 1, ack 1, win 229, options [nop,nop,TS val 2181395169 ecr 4022334050], length 0
13:05:22.647209 IP (tos 0x0, ttl 64, id 44768, offset 0, flags [none], proto TCP (6),
length 52)
    rcvm8.rcdevs.local.radius > 192.168.3.214.60210: Flags [F.], cksum 0x8935
(incorrect -> 0xee56), seq 1, ack 1, win 227, options [nop,nop,TS val 4022334051 ecr
2181395169], length 0
13:05:22.647350 IP (tos 0x10, ttl 64, id 3620, offset 0, flags [DF], proto TCP (6),
length 52)
    192.168.3.214.60210 > rcvm8.rcdevs.local.radius: Flags [F.], cksum 0xee52
(correct), seq 1, ack 2, win 229, options [nop,nop,TS val 2181395170 ecr 4022334051],
length 0
13:05:22.647389 IP (tos 0x0, ttl 64, id 44769, offset 0, flags [none], proto TCP (6),
length 52)
    rcvm8.rcdevs.local.radius > 192.168.3.214.60210: Flags [.], cksum 0x8935 (incorrect
-> 0xee53), seq 2, ack 2, win 227, options [nop,nop,TS val 4022334052 ecr 2181395170],
length 0
```

Example of a successful radius authentication:

```
[root@rcvm8 ~]# /opt/radiusd/bin/radtest test-user localhost:1812 testing123
Enter password: ********
Result: Success
Sent Access-Request Id 37 from 0.0.0.0:45386 to 127.0.0.1:1812 length 58 User-Name:
"test-user"
User-Password: "test1234"
NAS-Identifier: "RadTest"
Cleartext-Password: "test1234"

>>>> NOTE
Received Access-Accept Id 37 from 127.0.0.1:1812 to 127.0.0.1:45386 length 44 Reply-
Message: "Authentication success"
NOTE <<<<

[root@rcvm8 ~]#
```

```
[root@rcvm8 ~]# tcpdump -i any port 1812 -vv
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144
bytes
12:11:27.590459 IP (tos 0x0, ttl 64, id 13772, offset 0, flags [none], proto UDP (17),
length 86)
    localhost.55261 > localhost.radius: [bad udp cksum 0xfe55 -> 0x9692!] RADIUS,
length: 58
 Access-Request (1), id: 0xdd, Authenticator: b9cc3a8a669869cabaf009555a702080
   User-Name Attribute (1), length: 11, Value: test-user
     0x0000:  7465 7374 2d75 7365 72
   User-Password Attribute (2), length: 18, Value:
     0x0000:  79c7 cae6 2598 9c54 d83b 7c4a cfa8 3f7b
   NAS-Identifier Attribute (32), length: 9, Value: RadTest
     0x0000:  5261 6454 6573 74
12:11:27.631611 IP (tos 0x0, ttl 64, id 13792, offset 0, flags [none], proto UDP (17),
length 72)
    localhost.radius > localhost.55261: [bad udp cksum 0xfe47 -> 0x86a7!] RADIUS,
length: 44
 Access-Accept (2), id: 0xdd, Authenticator: 9901e9de1a3d690a092f860437e575ea
   Reply-Message Attribute (18), length: 24, Value: Authentication success
     0x0000:  4175 7468 656e 7469 6361 7469 6f6e 2073
     0x0010:  7563 6365 7373
```

8.3 Debug Mode

If the RADIUS Bridge is running and reachable on the correct address and port but authentications still fail, you can run it in debug mode using the below steps. This will show the detailed RADIUS authentication flow and any errors.

⚠ WARNING

Starting the OpenOTP RADIUS Bridge service in debug mode shows all passwords in clear text in the debug terminal!

```
[root@rcvm8 ~]# /opt/radiusd/bin/radiusd stop
Stopping OpenOTP RADIUS Bridge... Ok
[root@rcvm8 ~]# systemctl status radiusd
● radiusd.service - OpenOTP Radius Bridge
   Loaded: loaded (/usr/lib/systemd/system/radiusd.service; enabled; vendor preset:
disabled)

>>>> NOTE
   Active: failed (Result: signal) since Fri 2020-04-03 11:21:24 CEST; 3s ago
NOTE <<<<

  Process: 1024 ExecStart=/opt/radiusd/bin/radiusd start (code=exited,
status=0/SUCCESS)
 Main PID: 1635 (code=killed, signal=ABRT)

Apr 03 11:21:24 rcvm8.rcdevs.local systemd[1]: radiusd.service: Main process exited,
code=killed, status>
Apr 03 11:21:24 rcvm8.rcdevs.local systemd[1]: radiusd.service: Failed with result
'signal'.
[root@rcvm8 ~]# /opt/radiusd/bin/radiusd debug
Checking support for 32bit binaries... Ok
Checking server configuration... Ok
Starting OpenOTP RADIUS Bridge debug mode...
FreeRADIUS Version 3.0.20
Copyright (C) 1999-2019 The FreeRADIUS server project and contributors
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE
You may redistribute copies of FreeRADIUS under the terms of the
GNU General Public License
For more information about these matters, see the file named COPYRIGHT
Starting - reading configuration files ...
including dictionary file /opt/radiusd/lib/dictionaries/dictionary
including dictionary file /opt/radiusd/lib/dictionaries/dictionary.dhcp
including dictionary file /opt/radiusd/lib/dictionaries/dictionary.vqp
including configuration file /opt/radiusd/lib/radiusd.ini
including configuration file /opt/radiusd/conf/clients.conf
including configuration file /opt/radiusd/conf/radiusd.conf
main {
 security {
  user = "radiusd"
  group = "radiusd"
  allow_core_dumps = no
...
Listening on auth address * port 1812 bound to server default
Listening on auth proto tcp address * port 1812 bound to server default
```

```
Listening on auth address * port 1645 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on acct address * port 1646 bound to server default
Listening on status address * port 18120 bound to server default
Listening on command file /opt/radiusd/temp/radiusd.sock
Ready to process requests
```

8.4 Bad format or wrong RADIUS secret / Shared secret is incorrect

Logs example

- Radius Client (Radtest) IP is **192.168.3.54**
- Radius Bridge server IP is **192.168.3.64**

Radius client output :

```
[root@radius_cli ~]# /opt/radiusd/bin/radtest Administrator 192.168.3.64:1812
testing1234
Enter password: ********
(0) Reply verification failed: Received Access-Reject packet from home server
192.168.3.64 port 1812 with invalid Response Authenticator!  (Shared secret is
incorrect.)
```

Radius Bridge server debug output :

```
(0) Received Access-Request Id 186 from 192.168.3.54:45025 to 192.168.3.64:1812 length
62
(0)   User-Name = "Administrator"
(0)   User-Password = "T\\q\362\252ż\204\352:\177u;\241\205\243"
(0)   NAS-Identifier = "RadTest"
(0) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(0)   authorize {
(0) eap: No EAP-Message, not doing EAP
(0)     [eap] = noop
(0) pap: WARNING: No "known good" password found for the user.  Not setting Auth-Type
(0) pap: WARNING: Authentication will fail unless a "known good" password is available
(0)     [pap] = noop
rlm_openotp: Invalid "User-Password" attribute (bad format or wrong RADIUS secret)
(0)     [openotp] = invalid
(0)   } # authorize = invalid
(0) Invalid user: [Administrator] (from client any port 0)
(0) Using Post-Auth-Type Reject
(0) Post-Auth-Type sub-section not found.  Ignoring.
(0) Login incorrect: [Administrator] (from client any port 0)
(0) Sent Access-Reject Id 186 from 192.168.3.64:1812 to 192.168.3.54:45025 length 0
(0) Finished request
```

When that kind of error happens, it comes from a wrong Radius shared secret between Radius client and Radius Bridge.

Check the secret configured for this radius client on Radius Bridge server in `/opt/radiusd/conf/clients.conf` :

```
client 192.158.3.54 {
        ipaddr = 192.168.3.65
        secret = testing123
}
```

I used `testing1234` as secret on my radius client during the authentition. That is why it failed.

Note : If you use special characters in your Radius secret, then we advise to configure the secret in `clients.conf` between simple quotes like below :

```
client any {
        ipaddr = 192.168.3.65
        secret = 'testing123$!'
}
```

To use that secret with radtest, keep quotes (else `$` character will be interpreted by bash and it will fail) :

```
[root@radius_cli ~]# /opt/radiusd/bin/radtest Administrator 192.168.3.64:1812
'testing123$!'
Enter password: ********
(0) -: Expected Access-Accept got Access-Challenge
Result: Challenge
Session: 6a66626f47737239716371397a6e6f45
Enter your TOKEN password: 324032
Result: Success
Sent Access-Request Id 207 from 0.0.0.0:46922 to 192.168.3.64:1812 length 80 User-Name:
"Administrator"
User-Password: "324032"
State: 0x6a66626f47737239716371397a6e6f45 NAS-Identifier: "RadTest"
Cleartext-Password: "324032"
Received Access-Accept Id 207 from 192.168.3.64:1812 to 192.168.3.54:46922 length 44
Reply-Message: "Authentication success"
```

Radius Bridge output :

```
(0) Received Access-Request Id 124 from 192.168.3.54:41692 to 192.168.3.64:1812 length
62
(0)   User-Name = "Administrator"
(0)   User-Password = "password"
(0)   NAS-Identifier = "RadTest"
(0) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(0)   authorize {
(0) eap: No EAP-Message, not doing EAP
(0)     [eap] = noop
(0) pap: WARNING: No "known good" password found for the user.  Not setting Auth-Type
(0) pap: WARNING: Authentication will fail unless a "known good" password is available
(0)     [pap] = noop
(0)     [openotp] = ok
(0)   } # authorize = ok
(0) Found Auth-Type = OTP
(0) # Executing group from file /opt/radiusd/lib/radiusd.ini
(0)   Auth-Type OTP {
rlm_openotp: Found client ID attribute with value "RadTest"
rlm_openotp: Found source IP attribute with value ""
rlm_openotp: Found device ID attribute with value ""
rlm_openotp: Found client IP attribute with value ""
rlm_openotp: Sending openotpSimpleLogin request
rlm_openotp: OpenOTP authentication challenge
rlm_openotp: Reply message: Enter your TOKEN password
rlm_openotp: State: jfboGsr9qcq9znoE
rlm_openotp: Sending Access-Challenge
(0)     [openotp] = handled
(0)   } # Auth-Type OTP = handled
(0) Using Post-Auth-Type Challenge
(0) Post-Auth-Type sub-section not found.  Ignoring.
```

```
(0) Sent Access-Challenge Id 124 from 192.168.3.64:1812 to 192.168.3.54:41692 length 0
(0)   State := 0x6a66626f47737239716371397a6e6f45
(0)   Reply-Message := "Enter your TOKEN password"
(0)   Session-Timeout := 6908576
(0) Finished request
Waking up in 9.9 seconds.
(0) Cleaning up request packet ID 124 with timestamp +10
Ready to process requests
(1) Received Access-Request Id 207 from 192.168.3.54:46922 to 192.168.3.64:1812 length
80
(1)   User-Name = "Administrator"
(1)   User-Password = "324032"
(1)   State = 0x6a66626f47737239716371397a6e6f45
(1)   NAS-Identifier = "RadTest"
(1) session-state: No cached attributes
(1) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(1)   authorize {
(1) eap: No EAP-Message, not doing EAP
(1)     [eap] = noop
(1) pap: WARNING: No "known good" password found for the user.  Not setting Auth-Type
(1) pap: WARNING: Authentication will fail unless a "known good" password is available
(1)     [pap] = noop
(1)     [openotp] = ok
(1)   } # authorize = ok
(1) Found Auth-Type = OTP
(1) # Executing group from file /opt/radiusd/lib/radiusd.ini
(1)   Auth-Type OTP {
rlm_openotp: Found client ID attribute with value "RadTest"
rlm_openotp: Found source IP attribute with value ""
rlm_openotp: Found device ID attribute with value ""
rlm_openotp: Found client IP attribute with value ""
rlm_openotp: Found state attribute "State" with value "jfboGsr9qcq9znoE" (string)
rlm_openotp: Sending openotpChallenge request
rlm_openotp: OpenOTP authentication succeeded
rlm_openotp: Reply message: Authentication success
rlm_openotp: Sending Access-Accept
(1)     [openotp] = ok
(1)   } # Auth-Type OTP = ok
(1) Login OK: [Administrator] (from client any port 0)
(1) Sent Access-Accept Id 207 from 192.168.3.64:1812 to 192.168.3.54:46922 length 0
(1)   Reply-Message := "Authentication success"
(1) Finished request
```

For other Radius clients like Netscaler, F5, Palo Alto… the simple quotes must not be put in the secret field of your AAA Radius Server definition.

Below, the OpenOTP logs for the previous authentication in `/opt/webadm/logs/webadm.log`

```
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] New openotpSimpleLogin SOAP
request
```

```
request
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] > Username: Administrator
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] > Password: xxxxxxxx
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] > Client ID: RadTest
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] > Options: RADIUS,-U2F
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Registered openotpSimpleLogin
request
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Ignoring 2 memberof values for
user 'CN=Administrator,CN=Users,DC=yorcdevs,DC=eu' (out of domain group search base)
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Resolved LDAP user:
CN=Administrator,CN=Users,DC=yorcdevs,DC=eu
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Resolved LDAP groups: group
policy creator owners,domain admins,enterprise admins,schema admins,denied rodc
password replication group
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Using SQL server 'SQL Server'
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Using Session server 'Session
Server'
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Started transaction lock for
user
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Found user fullname:
Administrator
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Found 1 user emails:
Administrator@yorcdevs.eu
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Found 47 user settings:
LoginMode=LDAPOTP,OTPType=TOKEN,PushLogin=Yes,LockTimer=0,MaxTries=3,BlockTime=0,Challeng
1:HOTP-SHA1-6:QN06-
T1M,DeviceType=FIDO2,SMSType=Normal,SMSMode=Ondemand,MailMode=Ondemand,PrefetchExpire=10,
[1 Items]
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Found 9 user data:
TokenType,TokenKey,TokenState,TokenID,TokenSerial,Device1Type,Device1Name,Device1Data,Dev

[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Found 1 registered OTP token
(TOTP)
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Requested login factors: LDAP &
OTP
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] LDAP password Ok
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Authentication challenge
required
[2020-04-17 18:24:04] [192.168.3.64] [OpenOTP:85XTWHCO] Using Push server 'Push Server'
[2020-04-17 18:24:05] [192.168.3.64] [OpenOTP:85XTWHCO] Sent push notification for
token #1
[2020-04-17 18:24:05] [192.168.3.64] [OpenOTP:85XTWHCO] Waiting 27 seconds for mobile
response
[2020-04-17 18:24:32] [192.168.3.64] [OpenOTP:85XTWHCO] Started OTP authentication
session of ID jfboGsr9qcq9znoE valid for 90 seconds
[2020-04-17 18:24:32] [192.168.3.64] [OpenOTP:85XTWHCO] Sent login challenge response
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] New openotpChallenge SOAP
request
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] > Username: Administrator
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] > Session: jfboGsr9qcq9znoE
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] > OTP Password: xxxxxx
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] Registered openotpChallenge
request
```

```
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] Found authentication session
started 2020-04-17 18:24:04
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] Started transaction lock for
user
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] PUSH password Ok (token #1)
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] Updated user data
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] Sent terminate notification for
token #1
[2020-04-17 18:24:43] [192.168.3.64] [OpenOTP:85XTWHCO] Sent login success response
```

8.5 Wrong Client Definition

In this case, the IP address is for the client to connect is wrong. Please check the file `/opt/radiusd/conf/clients.conf`.

```
[root@rcvm8 ~]# /opt/radiusd/bin/radtest test-user localhost:1812 testing123
Enter password: ********

>>>> NOTE
(0) No reply from server for ID 141 socket 3
NOTE <<<<

Result: Error
[root@rcvm8 ~]#
```

```
[root@rcvm8 ~]# /opt/radiusd/bin/radiusd debug
...
Listening on auth address * port 1812 bound to server default
Listening on auth proto tcp address * port 1812 bound to server default
Listening on auth address * port 1645 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on acct address * port 1646 bound to server default
Listening on status address * port 18120 bound to server default
Listening on command file /opt/radiusd/temp/radiusd.sock
Ready to process requests

>>>> NOTE
Ignoring request to auth address * port 1812 bound to server default from unknown
client 127.0.0.1 port 39193 proto udp
NOTE <<<<

Ready to process requests
```

8.6 Missing MFA Enrolment

In this example, the radius authentication is set to LDAP and OTP. However, the user doesn't have yet a second factor enrolled.

```
[root@rcvm8 ~]# /opt/radiusd/bin/radtest test-user localhost:1812 testing123
Enter password: ********
(0) -: Expected Access-Accept got Access-Reject
Result: Failed
Sent Access-Request Id 89 from 0.0.0.0:58793 to 127.0.0.1:1812 length 58 User-Name:
"test-user"
User-Password: "test1234"
NAS-Identifier: "RadTest"
Cleartext-Password: "test1234"

>>>> NOTE
Received Access-Reject Id 89 from 127.0.0.1:1812 to 127.0.0.1:58793 length 81 Reply-
Message: "Account missing required data or MFA enrolment needed"
NOTE <<<<

Error-Cause: 26985728
[root@rcvm8 ~]#
```

```
[root@rcvm8 ~]# /opt/radiusd/bin/radiusd debug
...
Listening on auth address * port 1812 bound to server default
Listening on auth proto tcp address * port 1812 bound to server default
Listening on auth address * port 1645 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on acct address * port 1646 bound to server default
Listening on status address * port 18120 bound to server default
Listening on command file /opt/radiusd/temp/radiusd.sock
Ready to process requests
(0) Received Access-Request Id 89 from 127.0.0.1:58793 to 127.0.0.1:1812 length 58
(0)   User-Name = "test-user"
(0)   User-Password = "test1234"
(0)   NAS-Identifier = "RadTest"
(0) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(0)   authorize {
(0) eap: No EAP-Message, not doing EAP
(0)     [eap] = noop
(0) pap: WARNING: No "known good" password found for the user.  Not setting Auth-Type
(0) pap: WARNING: Authentication will fail unless a "known good" password is available
(0)     [pap] = noop
(0)     [openotp] = ok
(0)   } # authorize = ok
(0) Found Auth-Type = OTP
(0) # Executing group from file /opt/radiusd/lib/radiusd.ini
(0)   Auth-Type OTP {
rlm_openotp: Found client ID attribute with value "RadTest"
rlm_openotp: Found source IP attribute with value ""
rlm_openotp: Found device ID attribute with value ""
```

```
rlm_openotp: Found client IP attribute with value ""
rlm_openotp: Sending openotpSimpleLogin request
rlm_openotp: OpenOTP authentication failed

>>>> NOTE
rlm_openotp: Reply message: Account missing required data or MFA enrolment needed
NOTE <<<<

rlm_openotp: Sending Access-Reject
(0)     [openotp] = reject
(0)   } # Auth-Type OTP = reject
(0) Failed to authenticate the user
(0) Using Post-Auth-Type Reject
(0) Post-Auth-Type sub-section not found.  Ignoring.
(0) Login incorrect: [test-user] (from client any port 0)
(0) Sent Access-Reject Id 89 from 127.0.0.1:1812 to 127.0.0.1:58793 length 0
(0)   Reply-Message := "Account missing required data or MFA enrolment needed"
(0)   Error-Cause := 26985728
(0) Finished request
Waking up in 9.9 seconds.
(0) Cleaning up request packet ID 89 with timestamp +8
Ready to process requests
```

```
[root@rcvm8 ~]# tail -f /opt/webadm/logs/webadm.log
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] New openotpSimpleLogin SOAP
request
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] > Username: test-user
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] > Password: xxxxxxxx
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] > Client ID: RadTest
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] > Options: RADIUS,-U2F
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] Registered openotpSimpleLogin
request
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] Resolved LDAP user: cn=test-
user,o=Root
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] Started transaction lock for user
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] Found user fullname: test-user
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] Found 46 user settings:
LoginMode=LDAPOTP,OTPType=TOKEN,ChallengeMode=Yes,ChallengeTimeout=90,OTPLength=6,Mobile1
1:HOTP-SHA1-6:QN06-
T1M,DeviceType=FIDO2,SMSType=Normal,SMSMode=Ondemand,MailMode=Ondemand,PrefetchExpire=10,


>>>> NOTE
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] User has no OTP token registered
NOTE <<<<

[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] No usable login method found
[2020-04-03 11:54:06] [127.0.0.1] [OpenOTP:2WL6CTJJ] Sent failure response
```

8.7 Incorrect Protocol

Only PAP or EAP-TTLS authentication is supported for password authentication. CHAP and similar password protocols using hashed values are not supported. If you Windows clients such as Windows integrated VPN client, be sure to configure the VPN client and server with PAP authentication.

```
[root@rcvm8 ~]# /opt/radiusd/bin/radiusd debug
...
Listening on auth address * port 1812 bound to server default
Listening on auth proto tcp address * port 1812 bound to server default
Listening on auth address * port 1645 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on acct address * port 1646 bound to server default
Listening on status address * port 18120 bound to server default
Listening on command file /opt/radiusd/temp/radiusd.sock
Ready to process requests
(0)   NAS-IP-Address = 10.10.10.10
(0)   NAS-Port = 1
(0)   User-Name = "test-user"

>>>> NOTE
(0)   CHAP-Challenge = 0x11111111111111111111111
(0)   CHAP-Password = 0x22222222222222222222**
NOTE <<<<

(0)   Message-Authenticator = 0x3333333333333333333333
(0) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(0)   authorize {
(0) eap: No EAP-Message, not doing EAP
(0)     [eap] = noop
(0) pap: WARNING: No "known good" password found for the user.  Not setting Auth-Type
(0) pap: WARNING: Authentication will fail unless a "known good" password is available
(0)     [pap] = noop
rlm_openotp: Invalid "User-Password" attribute (bad format or wrong RADIUS secret)
...
```

8.8 Radius Returned Attributes

8.8.1 Invalid RADIUS return attributes

It's possible to configure radius retuned attributes through WebADM GUI for specific users, groups or clients applications. Please refer to Radius Attributes documentation for how to configure them. RADIUS return attributes must comply with the RADIUS dictionaries stored in `/opt/radiusd/lib/dictionaries/`. If they do not, the authentication will fail. In the example below, RADIUS Bridge receives return attribute `ASA-VLAN="string"`, which is not correct as the attribute is defined as integer.

```
[root@rcvm8 ~]# /opt/radiusd/bin/radiusd debug
...
rlm_openotp: OpenOTP authentication succeeded

>>>> NOTE
rlm_openotp: Reply Data: ASA-VLAN="string"
rlm_openotp: Invalid Reply Data (invalid value-pairs format or attribute not in
dictionary)
NOTE <<<<

(3)     [openotp] = fail
(3)   } # Auth-Type OTP = fail
(3) Failed to authenticate the user
(3) Using Post-Auth-Type Reject
(3) Post-Auth-Type sub-section not found.  Ignoring.
(3) Login incorrect: [test] (from client any port 0)
(3) Sent Access-Reject Id 52 from 127.0.0.1:1812 to 127.0.0.1:34295 length 0
(3) Finished request
```

8.8.2 Check Radius Returned Attributes

For this test, I configured `Citrix-User-Groups` as Radius returned attribute with a mapping to `memberof` attribute of my Administrator account.

Radius Client output :

```
[root@radius_cli ~]# /opt/radiusd/bin/radtest Administrator 192.168.3.64:1812
'testing123$!'
Enter password: ********
Result: Success
Sent Access-Request Id 55 from 0.0.0.0:60026 to 192.168.3.64:1812 length 62 User-Name:
"Administrator"
User-Password: "password"
NAS-Identifier: "RadTest"
Cleartext-Password: "password"
Received Access-Accept Id 55 from 192.168.3.64:1812 to 192.168.3.54:60026 length 410
Citrix-User-Groups: "CN=Organization Management,OU=Microsoft Exchange Security
Groups,DC=yorcdevs,DC=eu"
Citrix-User-Groups: "CN=Group Policy Creator Owners,CN=Users,DC=yorcdevs,DC=eu"
Citrix-User-Groups: "CN=Domain Admins,CN=Users,DC=yorcdevs,DC=eu"
Citrix-User-Groups: "CN=Enterprise Admins,CN=Users,DC=yorcdevs,DC=eu"
Citrix-User-Groups: "CN=Schema Admins,CN=Users,DC=yorcdevs,DC=eu"
Citrix-User-Groups: "CN=Administrators,CN=Builtin,DC=yorcdevs,DC=eu"
Reply-Message: "Authentication success"
```

As you can see, groups of my Administrator account are well returned.

Radius Bridge output :

```
(2) Received Access-Request Id 55 from 192.168.3.54:60026 to 192.168.3.64:1812 length
62
(2)   User-Name = "Administrator"
(2)   User-Password = "password"
(2)   NAS-Identifier = "RadTest"
(2) # Executing section authorize from file /opt/radiusd/lib/radiusd.ini
(2)   authorize {
(2) eap: No EAP-Message, not doing EAP
(2)     [eap] = noop
(2) pap: WARNING: No "known good" password found for the user.  Not setting Auth-Type
(2) pap: WARNING: Authentication will fail unless a "known good" password is available
(2)     [pap] = noop
(2)     [openotp] = ok
(2)   } # authorize = ok
(2) Found Auth-Type = OTP
(2) # Executing group from file /opt/radiusd/lib/radiusd.ini
(2)   Auth-Type OTP {
rlm_openotp: Found client ID attribute with value "RadTest"
rlm_openotp: Found source IP attribute with value ""
rlm_openotp: Found device ID attribute with value ""
rlm_openotp: Found client IP attribute with value ""
rlm_openotp: Sending openotpSimpleLogin request
rlm_openotp: OpenOTP authentication succeeded
rlm_openotp: Reply Data: Citrix-User-Groups="CN=Organization Management,OU=Microsoft
Exchange Security Groups,DC=yorcdevs,DC=eu",Citrix-User-Groups="CN=Group Policy Creator
Owners,CN=Users,DC=yorcdevs,DC=eu",Citrix-User-Groups="CN=Domain
Admins,CN=Users,DC=yorcdevs,DC=eu",Citrix-User-Groups="CN=Enterprise
Admins,CN=Users,DC=yorcdevs,DC=eu",Citrix-User-Groups="CN=Schema
Admins,CN=Users,DC=yorcdevs,DC=eu",Citrix-User-
Groups="CN=Administrators,CN=Builtin,DC=yorcdevs,DC=eu"
rlm_openotp: Reply message: Authentication success
rlm_openotp: Sending Access-Accept
(2)     [openotp] = ok
(2)   } # Auth-Type OTP = ok
(2) Login OK: [Administrator] (from client any port 0)
(2) Sent Access-Accept Id 55 from 192.168.3.64:1812 to 192.168.3.54:60026 length 0
(2)   Citrix-User-Groups = "CN=Organization Management,OU=Microsoft Exchange Security
Groups,DC=yorcdevs,DC=eu"
(2)   Citrix-User-Groups = "CN=Group Policy Creator Owners,CN=Users,DC=yorcdevs,DC=eu"
(2)   Citrix-User-Groups = "CN=Domain Admins,CN=Users,DC=yorcdevs,DC=eu"
(2)   Citrix-User-Groups = "CN=Enterprise Admins,CN=Users,DC=yorcdevs,DC=eu"
(2)   Citrix-User-Groups = "CN=Schema Admins,CN=Users,DC=yorcdevs,DC=eu"
(2)   Citrix-User-Groups = "CN=Administrators,CN=Builtin,DC=yorcdevs,DC=eu"
(2)   Reply-Message := "Authentication success"
(2) Finished request
```

Below, the OpenOTP logs for the previous authentication. You can see once the authentication factors are validated by OpenOTP, OpenOTP return the attribute configured on my Administrator account and then the log regarding radius returned attribut appears : `Returning 6 RADIUS reply attributes`.

```
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] New openotpSimpleLogin SOAP
request
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] > Username: Administrator
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] > Password: xxxxxxxx
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] > Client ID: RadTest
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] > Options: RADIUS,-U2F
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Registered openotpSimpleLogin
request
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Ignoring 2 memberof values for
user 'CN=Administrator,CN=Users,DC=yorcdevs,DC=eu' (out of domain group search base)
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Resolved LDAP user:
CN=Administrator,CN=Users,DC=yorcdevs,DC=eu
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Resolved LDAP groups: group
policy creator owners,domain admins,enterprise admins,schema admins,denied rodc
password replication group
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Started transaction lock for
user
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Found user fullname:
Administrator
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Found 1 user emails:
Administrator@yorcdevs.eu
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Found 47 user settings:
LoginMode=LDAPOTP,OTPType=TOKEN,PushLogin=Yes,LockTimer=0,MaxTries=3,BlockTime=0,Challeng
1:HOTP-SHA1-6:QN06-
T1M,DeviceType=FIDO2,SMSType=Normal,SMSMode=Ondemand,MailMode=Ondemand,PrefetchExpire=10,
[1 Items]
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Found 10 user data:
LastOTP,TokenType,TokenKey,TokenState,TokenID,TokenSerial,Device1Type,Device1Name,Device1

[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Last OTP expired 2020-04-17
18:29:43
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Found 1 registered OTP token
(TOTP)
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Requested login factors: LDAP &
OTP
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] LDAP password Ok
[2020-04-17 18:50:45] [192.168.3.64] [OpenOTP:4JKBFJ4C] Authentication challenge
required
[2020-04-17 18:50:46] [192.168.3.64] [OpenOTP:4JKBFJ4C] Sent push notification for
token #1
[2020-04-17 18:50:46] [192.168.3.64] [OpenOTP:4JKBFJ4C] Waiting 27 seconds for mobile
response
[2020-04-17 18:50:56] [192.168.3.56] [OpenOTP:4JKBFJ4C] Received mobile authentication
response from 192.168.3.1
[2020-04-17 18:50:56] [192.168.3.56] [OpenOTP:4JKBFJ4C] > Session: cupcbM2KWdmcAxjF
[2020-04-17 18:50:56] [192.168.3.56] [OpenOTP:4JKBFJ4C] > Password: 16 Bytes
[2020-04-17 18:50:56] [192.168.3.56] [OpenOTP:4JKBFJ4C] Found authentication session
```

```
started 2020-04-17 18:50:45
[2020-04-17 18:50:56] [192.168.3.56] [OpenOTP:4JKBFJ4C] PUSH password Ok (token #1)
[2020-04-17 18:50:56] [192.168.3.64] [OpenOTP:4JKBFJ4C] Returning 6 RADIUS reply
attributes
[2020-04-17 18:50:56] [192.168.3.64] [OpenOTP:4JKBFJ4C] Updated user data
[2020-04-17 18:50:56] [192.168.3.64] [OpenOTP:4JKBFJ4C] Sent login success response
```

Appendix A: Password Modes

The OpenOTP Web Service (i.e. the OpenOTP main API) supports multiple password check mechanisms. The supported OpenOTP user login modes are:

0 - AUTO: This the default operating mode where RadiusBridge let OpenOTP handle the passwords automatically. This mode uses OpenOTP v1.1 SimpleLogin API. This mode is highly recommended for common integrations.

1 - LDAP: OpenOTP needs to checks the user LDAP password only. In this mode, the client system must provide the LDAP password in the openotpLogin request.

2 - OTP: OpenOTP needs to checks the user OTP password only. In this mode, the client system can provide the OTP password in the openotpLogin request or no password at all. If no password is provided, then OpenOTP will issue a Challenge-Response and the client application will have to provide the OTP password in an openotpChallenge request (in a second OpenOTP request).

3 - LDAPOTP: OpenOTP needs to checks both LDAP and OTP passwords. In this mode, the client system can provide the LDAP and OTP password in the openotpLogin request or only the LDAP password. If only the LDAP is provided, then OpenOTP will issue a Challenge-Response and the client system will have to provide the OTP password in an openotpChallenge request (in a second OpenOTP request).

So as a summary, in WebADM, you can have users which are configured with Login Modes: LDAP or OTP or LDAPOTP. There is a default mode configured in your OpenOTP Web Service application and it can be re-defined per group, user or client application policies.

Then you have secondary systems like RADIUS (with OpenOTP Radius Bridge), PAM or other integrated systems where you are more limited in terms of password functionalities due to the specificities of these technologies. So you must consider OpenOTP is a more generic authentication framework and RADIUS/PAM are OpenOTP authentication subsystems which have specific password capabilities.

In RADIUS or PAM, you can use password modes 1, 2, 3 or 4 (with the password_mode setting of the RADIUS/PAM configuration).

1. RADIUS/PAM password mode 1: This is the default mode where RADIUS/PAM sends the LDAP password and expects a positive, negative or challenge response from OpenOTP. Challenge mode is used if the OTP is required and only the LDAP password was sent. This mode supports both LDAP and LDAPOTP user login modes. If LDAP only is required, then the response is positive or negative and if OTP is required, then the response is a challenge or negative. In the challenge, RADIUS will return a RADIUS challenge-response to the client. But RADIUS challenge is not supported by every client. So OpenOTP Radius Bridge supports some concatenated passwords modes as described below. With PAM integrations, some PAM services such as OpenSSH support challenge mode but some other such as FTP do not. With password_mode 1, your OpenOTP users must be configured with LDAPOTP or LDAP Login Mode. It must also be noted that some OpenOTP login methods such as SMS or email work only in

challenge mode as a first request must trigger the SMS/mail send.

2. RADIUS/PAM password mode 2: This mode is used for logins with OTP password only. Then RADIUS/PAM sends the OTP directly and no LDAP password. The OpenOTP user must also be configured with OTP Login Mode.

3. RADIUS/PAM password mode 3 and 4: These modes are used for password concatenations. This is useful if the system does not support the challenge and you want LDAP+OTP. RADIUS/PAM sends LDAP+OTP passwords concatenated and different forms of concatenation are possible (with a separator character or defined OTP length). This mode supports users with LDAPOTP login mode but it supports OTP only too. When the separator is missing or length is equal to the defined OTP length, then RADIUS/PAM assumes only the OTP password was provided and does not send the LDAP password. Password modes 3 and 4 can also be used when the OpenOTP users are configured with LDAPOTP or OTP Login Modes. In RADIUS/PAM, it is important to notice that you can play with the password_mode settings but also with the user_settings setting. The user_settings allows passing some OpenOTP user configurations directly from the client system. For example, you can set user_settings="OpenOTP.LoginMode=OTP" to tell OpenOTP to work in OTP mode in a PAM configuration for FTP. And this, even when users are configured with LDAPOTP Login Mode.

4. RADIUS/PAM password mode 5: This mode supports the concatenation of the username and the OTP password. It has been added for mainly for Yubikey when concatenation is required (no challenge) and the RADIUS client has a limitation in the length of the RADIUS password attribute.