BaroPAM Guide(RADIUS)

Index

1. RADIUS authentication

1.1 Introduction

RADIUS (Remote Authentication Dial In User Service) is a networking protocol that manages centralized authentication, authorization, and accounting (AAA, Accounting takes charge of various post-processing after authentication and authorization) for users to connect to the network and receive network services. provides RADIUS is a server access authentication and accounting protocol established in 1991 by Livingston Enterprises, Inc. developed in It was later listed as an IETF standard.

Because it has a wide range of support and can be used in a ubiquitous environment, it is often used by ISPs and corporations to manage Internet or intranet access, wireless network authentication, and integrated mail services. Used in modems, wireless access points, digital subscriber lines, virtual private networks, TCP and UDP ports, web servers, etc.

RADIUS is a client and server protocol that operates at the application layer and is transmitted over the user datagram protocol. Remote access servers, virtual private networks, port authentication on network switches, and network access servers (NAS) all have components that communicate with a RADIUS server. RADIUS is often the basis for IEEE 802.1X authentication.

RADIUS servers often run as background processes on Linux/Unix systems or Windows servers.

The process flow, the RADIUS authentication process, is relatively simple but important to understand.

1. The application client attempts to connect to the application server and provides credentials (**login-ID and password**) along with a **OTA key**.

2. The application server receives this information, looks up the user in the local catalog, and verifies that the authentication type is PAM.

3. Then decide where to authenticate external users.

4. Using the RADIUS connection information, the application server forwards credential details to the RADIUS server.

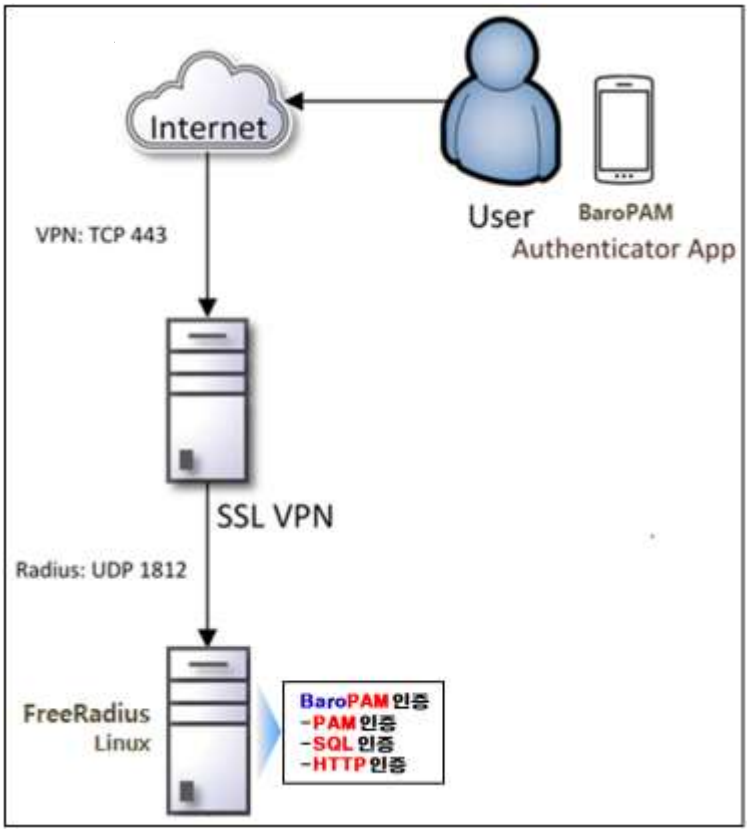
5. The RADIUS server first does "**Primary Authentication**" with **Login-ID/Password using PAM**, which can be PAM, HTTP, SQL, Active Directory, LDAP service, etc.

6. If validity is confirmed, the RADIUS server performs "**Secondary Authentication**" with the **OTA key** with the BaroPAM module, which is a secondary authentication service.

7. If validation is confirmed, the RADIUS server sends an "**Access-Accept**" response back to the application server, then accepts and completes the connection.

Consistent and accurate timing is a key requirement for the operation of the proposed solution. When using a RADIUS host and a time-based One-Time Authentication key, the device with the **BaroPAM** service and **BaroPAM** app installed must have consistent time, so NTP (Network Time Protocol) must be set.

1.2 Configuration architecture



1.3 Component

-Rocky linuxx
-FreeRADIUS
-BaroPAM PAM Library, Service, & APP
-Pluggable Authentication Module (PAM)

1.4 Firewall Requirements

1) User -> NAS
-TCP 443: SSL VPN

2) NAS -> RADIUS
-UDP 1812: RADIUS

2. FreeRADIUS installation and setup

2.1 FreeRADIUS installation

```
[root@localhost ~]# dnf -y install freeradius freeradius-utils
```

When trying to uninstall the installed FreeRADIUS ➔ **dnf -y erase freeradius freeradius-utils**

If MySQL/MariaDB is linked, the MySQL/MariaDB module must be additionally installed.

```
[root@localhost ~]# dnf install -y freeradius-mysql
```

After installing FreeRADIUS, you must create a certificate for EAP.

```
$ cd  /etc/raddb/certs
$ ./bootstrap
```

If you do not create a certificate for EAP, the following error occurs.

```
Failed reading private key file /etc/raddb/certs/server.pem
:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt
rlm_eap_tls: Failed initializing SSL context
rlm_eap (EAP): Failed to initialise rlm_eap_tls
/etc/raddb/mods-enabled/eap[17]: Instantiation failed for module "eap"
```

2.2 FreeRADIUS settings

1) PAM, cURL authentication

Change both user and group to **root** as follows.

```
[root@localhost ~]# vi /etc/raddb/radiusd.conf
#user = radiusd
#group = radiusd
user = root
group = root
```

Using FreeRADIUS requires running as **root** to access BaroPAM in the user's home directory.

For easier troubleshooting, enable additional logging by editing **/etc/raddb/radiusd.conf** in the log directive.

```
auth = yes
auth_badpass = yes
auth_goodpass = yes
```

To enable PAM you need to edit **/etc/raddb/sites-enabled/default**. Uncomment the #PAM line to enable

PAM.

```
[root@localhost ~]# vi /etc/raddb/sites-enabled/default

#Pluggable Authentication Modules.
pam
```

Remove the comment (#) to enable FreeRADIUS login "**auth_log**" and "**reply_log**".

```
[root@localhost ~]# vi /etc/raddb/sites-enabled/default

#
        #  If you want to have a log of authentication requests,
        #  un-comment the following line, and the 'detail auth_log'
        #  section, above.
#       auth_log
        auth_log

        #
        #  If you want to have a log of authentication replies,
        #  un-comment the following line, and the 'detail reply_log'
        #  section, above.
#       reply_log
        reply_log
```

Create a soft link for PAM in **/etc/raddb/mods-enabled/** to enable PAM as follows.

```
[root@localhost ~]# ln -s /etc/raddb/mods-available/pam /etc/raddb/mods-enabled/
```

Change Ipv4addr and secret as follows.

```
[root@localhost ~]# vi /etc/raddb/clients.conf

client localhost {
    ipaddr = 127.0.0.1
    ipv4addr = *    # any.  127.0.0.1 == localhost
    secret = baropam
    require_message_authenticator = no
    nas_type = other
}
```

Change the authentication type to PAM as follows.

```
[root@localhost ~]# vi /etc/raddb/users

DEFAULT Group == "disabled", Auth-Type := Reject
Reply-Message = "Your account has been disabled."
DEFAULT Auth-Type := PAM
```

2) SQL authentication

Create a soft link for SQL in **/etc/raddb/mods-enabled/** to enable SQL as follows.

```
[root@localhost ~]# ln -s /etc/raddb/mods-available/sql /etc/raddb/mods-enabled/
```

Change Ipv4addr and secret as follows.

```
[root@localhost ~]# vi /etc/raddb/clients.conf

client localhost {
    ipaddr = 127.0.0.1
    ipv4addr = *    # any.  127.0.0.1 == localhost
    secret = baropam
    require_message_authenticator = no
    nas_type = other
}
```

Now configure FreeRADIUS to use MySQL/MariaDB. Edit the **/etc/raddb/mods-available/sql** file as follows.

```
driver = "rlm_sql_null" ==> driver = "rlm_sql_mysql"
dialect = "sqlite"      ==> dialect = "mysql"
```

When configuring FreeRADIUS to work with MySQL/MariaDB, the MySQL configuration assumes the use of TLS by default. Comment out the TLS section as we are not using an SSL certificate here.

```
mysql {
# If any of the files below are set, TLS encryption is enabled
# tls {
# ca_file = "/etc/ssl/certs/my_ca.crt"
# ca_path = "/etc/ssl/certs/"
# certificate_file = "/etc/ssl/certs/private/client.crt"
# private_key_file = "/etc/ssl/certs/private/client.key"
# cipher = "DHE-RSA-AES256-SHA:AES128-SHA"
# tls_required = yes
# tls_check_cert = no
# tls_check_cert_cn = no
#}
# If yes, (or auto and libmysqlclient reports warnings are
# available), will retrieve and log additional warnings from
# the server if an error has occured. Defaults to 'auto'
warnings = auto
```

Uncomment the server, port, login and password and change some values to set the MySQL/MariaDB connection information.

```
# Connection info:
#
server = "localhost"
port = 3306
login = "radius"
password = "baropam"
# Database table configuration for everything except Oracle
radius_db = "radius"
```

```
read_clients = yes
```

Change the group permission of the edited **/etc/raddb/mods-available/sql** file to radiusd.

```
chgrp -h radiusd /etc/raddb/mods-enabled/sql
```

3) allow firewall

Firewalld must be configured to allow RADIUS packets.

RADIUS uses port **1812** for authentication and port **1813** for accounting, so you need to allow traffic on these ports. Installing FreeRADIUS adds a configuration to your firewall, so allow RADIUS traffic and run the following command.

```
[root@localhost ~]# firewall-cmd --permanent --zone=public --add-port=1812/udp
success
[root@localhost ~]# firewall-cmd --permanent --zone=public --add-port=1813/udp
success
```

or

```
[root@localhost ~]# firewall-cmd --add-service=radius --permanent
success
```

You need to reload the firewall for the changes to take effect, then run the following command.

```
 [root@localhost ~]# firewall-cmd --reload
success
```

Note) Run firewall firewall-cmd installation

```
[root@localhost ~]# firewall-cmd
usage: see firewall-cmd man page
No option specified.

[root@localhost ~]# firewall-cmd --zone=public --permanent --add-port=21/tcp
FirewallD is not running


1. Check if it works

1-1. Method
[root@localhost ~]# systemctl status firewalld
* firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:firewalld(1)

1-2. Method
[root@localhost ~]# firewall-cmd --state
not running
```

```
2. Check firewalld installation
[root@localhost ~]# yum list installed firewalld
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: ftp.daumkakao.com
 * epel: ftp.riken.jp
 * extras: ftp.daumkakao.com
 * updates: ftp.daumkakao.com
 * webtatic: sp.repo.webtatic.com
Installed Packages
firewalld.noarch                0.4.4.4-6.el7                @base


3. Delete package
[root@localhost ~]# yum remove firewalld.noarch


4. firewall installation
[root@localhost ~]# yum install firewalld


5. firewall run
[root@localhost my.cnf.d]# systemctl start firewalld


6. Check firewall status
[root@localhost my.cnf.d]# systemctl status firewalld
* firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2018-01-30 17:12:19 KST; 4s ago
     Docs: man:firewalld(1)
 Main PID: 21620 (firewalld)
   CGroup: /system.slice/firewalld.service
           `-21620 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid
```

4) Create RADIUS service for auto start

Create a RADIUS service for automatic startup.

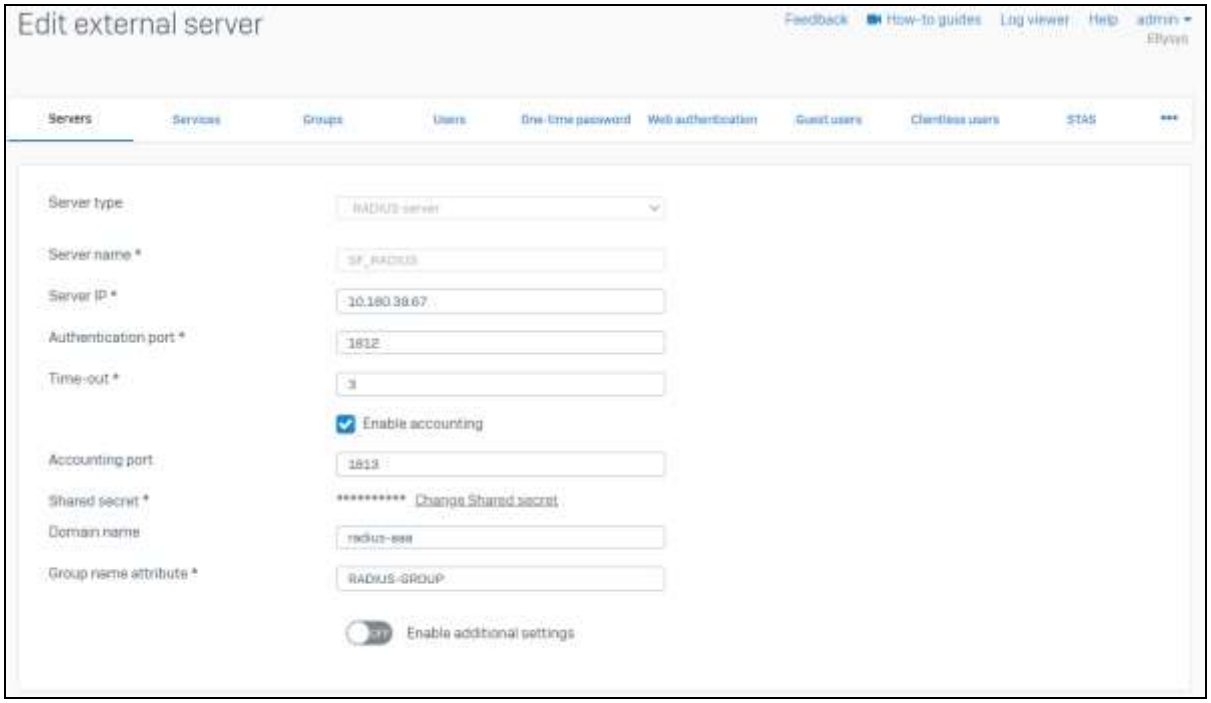
```
[root@localhost ~]# systemctl enable radiusd.service
Created symlink /etc/systemd/system/multi-user.target.wants/radiusd.service ->
/usr/lib/systemd/system/radiusd.service.
```

5) Authentication server settings

Ex) Authentication of OpenVPN: RADIUS setting screen

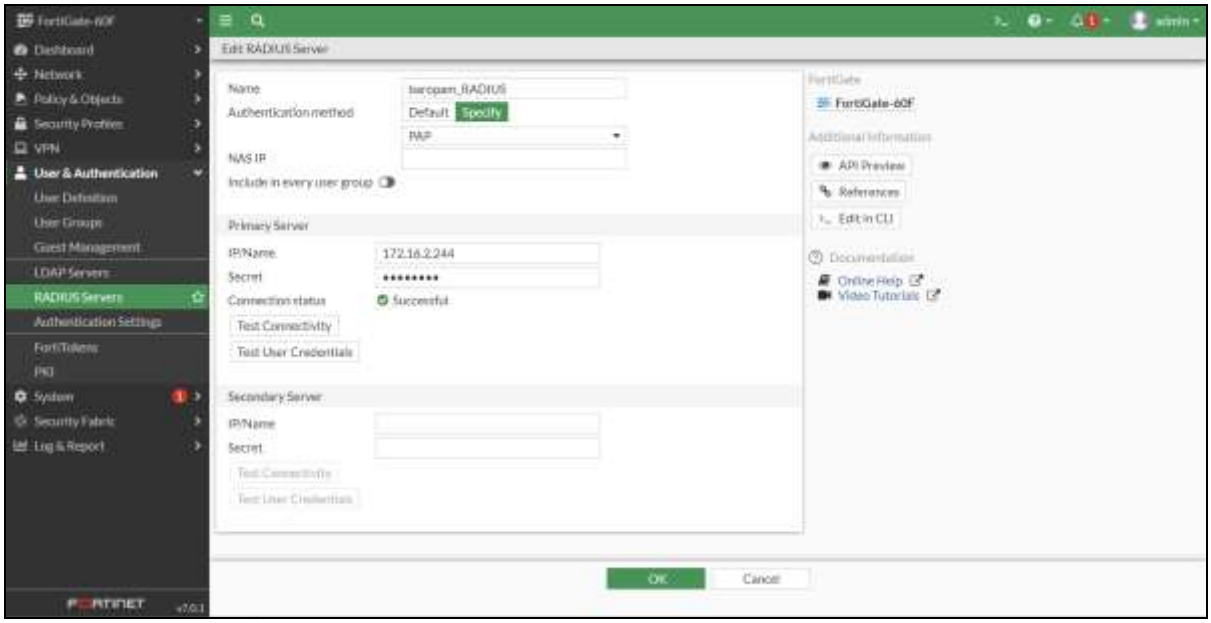
* Enable RADIUS authentication: Yes, Authentication Method: PAP, Shared Secret: baropam
(PAP, Ppassword authentication protocolassword Authentication Protocol)

Ex) Authentication server setting screen of Sophos SSLVPN



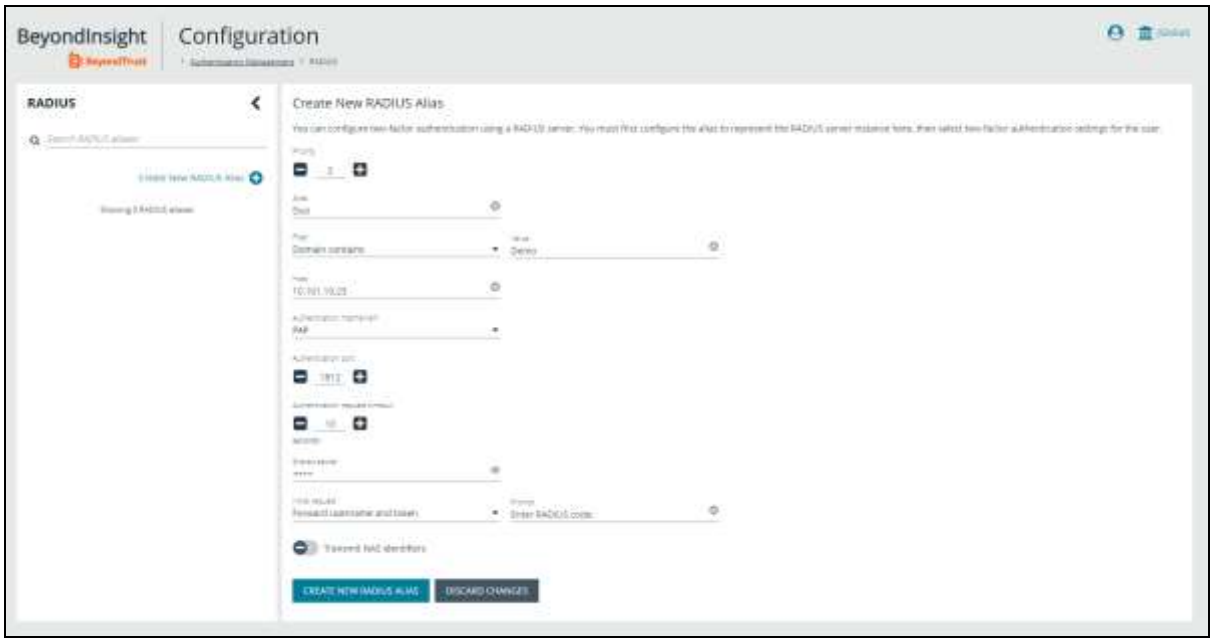
* Shared Secret: baropam

Ex) Authentication server setting screen of FotiGate SSLVPN



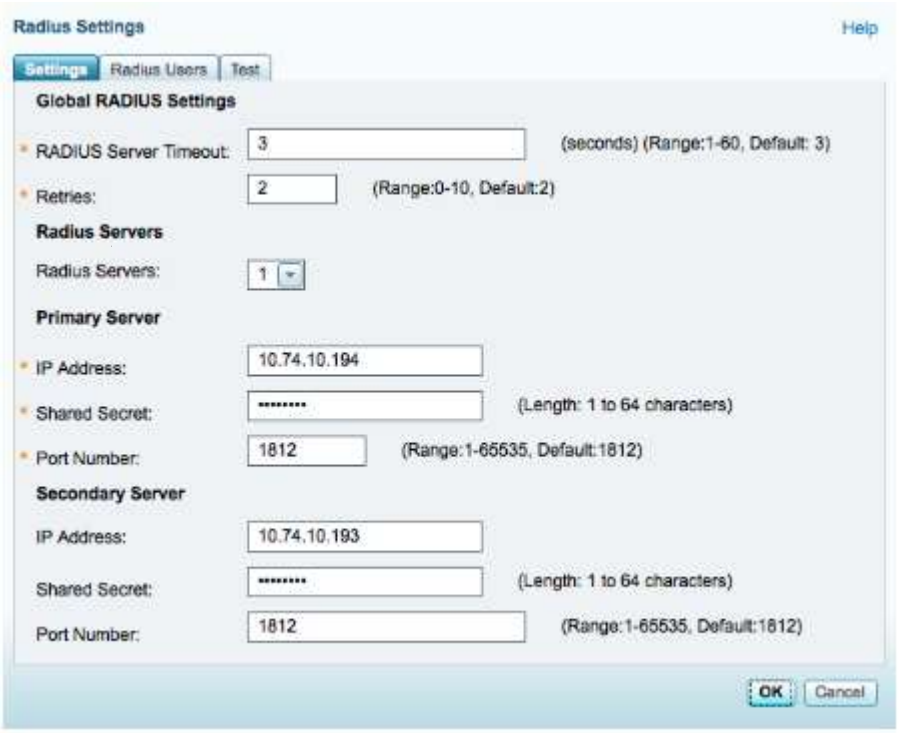
* Authentication Mechanism: Select PAP, Secret: baropam

Ex) Authentication server setting screen of Beyondtrust Password Safe



* Authentication Mechanism: PAP, Initial Request: Forward User Name and token, Secret: baropam

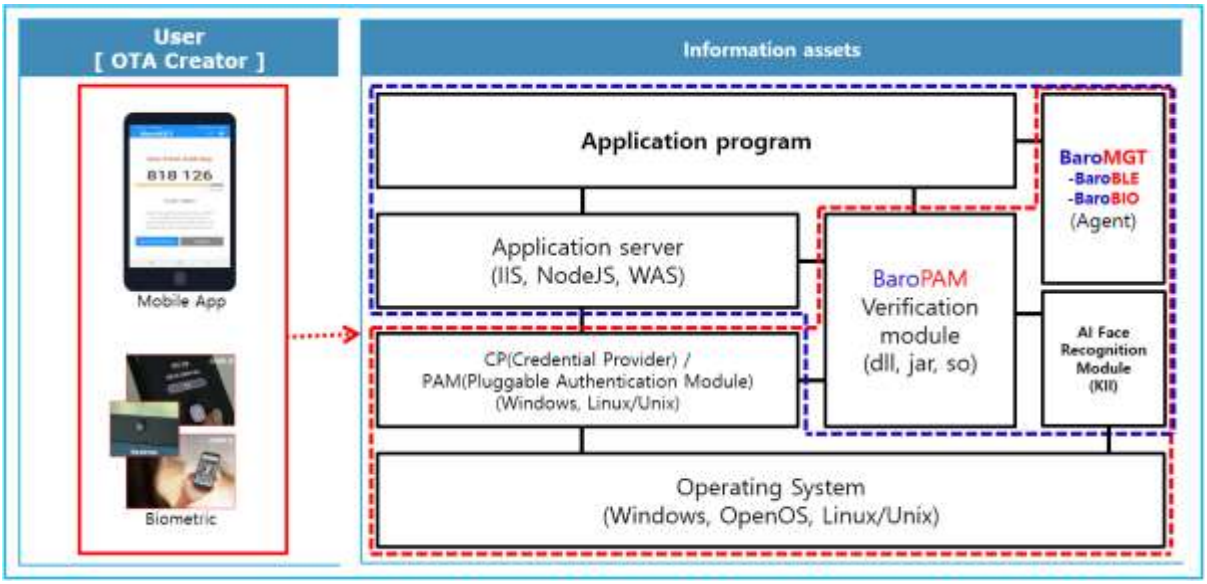
Ex) Cisco SSLVPN Authentication: RADIUS settings screen



* Shared Secret: baropam

3. BaroPAM installation and setup

The BaroPAM solution is a zero trust security model based on the Pluggable Authentication Module (PAM) method that anyone can easily and immediately apply to various operating systems and applications that require secondary authentication (additional authentication) to enhance the security of information assets. It is a 3-step authentication solution with biometrics optimized for security.



3.1 Preparation before installing BaroPAM

To use the PAM module, the PAM package must be installed by default. To check the installation, run the following command. If it is not installed, use the command "**dnf install pam**" for Redhat series and "**sudo apt-get install pam**" for others.

```
[root]# rpm -qa | grep pam
pam_smb-1.1.7-7.2.1
pam_passwdqc-1.0.2-1.2.2
pam-0.99.6.2-14.el5_11
pam_krb5-2.2.14-22.el5
pam-devel-0.99.6.2-14.el5_11
pam_ccreds-3-5
pam_smb-1.1.7-7.2.1
pam_pkcs11-0.5.3-26.el5
pam-devel-0.99.6.2-14.el5_11
pam_passwdqc-1.0.2-1.2.2
pam-0.99.6.2-14.el5_11
pam_ccreds-3-5
pam_krb5-2.2.14-22.el5
pam_pkcs11-0.5.3-26.el5
```

In the case of Redhat series, "**Selinux**" is an abbreviation of "**Security Enhanced Linux**" and provides a more excellent security policy than the basic Linux. If it is so outstanding that it is

activated, a part where BaroPAM cannot be blocked due to security problems occurs (**Failed to open tmp secret file "/usr/baropam/.baro_auth~" [Permission denied]**). So, if possible, most of them are disabled (**SELINUX=enforcing → disabled**).

```
[root] /etc > vi /etc/sysconfig/selinux
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - SELinux is fully disabled.
SELINUX=disabled
# SELINUXTYPE= type of policy in use. Possible values are:
#       targeted - Only targeted network daemons are protected.
#       strict - Full SELinux protection.
SELINUXTYPE=targeted

# SETLOCALDEFS= Check local definition changes
SETLOCALDEFS=0
```

It doesn't take effect right away and requires a reboot to take effect.

If you want to apply the changes only to the currently connected terminal without rebooting, run the following command.

```
[root] /etc > /usr/sbin/setenforce 0
```

To download and install the BaroPAM authentication module, connect with the **root** account and create a directory (/usr/baropam) to download and install the module as follows.

```
[root]# mkdir /usr/baropam
```

Grant permissions (read, write, execute) of the directory to download and install the BaroPAM module as follows.

```
[root]# chmod -R 777 /usr/baropam
```

3.2 Download BaroPAM installation module

After accessing the BaroPAM authentication module with the **root** account, move to the directory (/usr/baropam) to download and install the module, and download the module as follows.

```
[root] /usr/baropam > wget http://nuriapp.com/download/libpam_baro_auth-x.x.tar
```

When the download of the BaroPAM authentication module is complete, the **tar** file is decompressed as follows.

```
[root] /usr/baropam > tar -xvf libpam_baro_auth-x.x.tar
```

When the BaroPAM authentication module is unzipped, the following BaroPAM related modules are created in the baropam directory.

```
[root] /usr/baropam > ls -al
합계 180
drwxrwxrwx  7 root   root   4096  8월 23 09:59 .
drwxr-xr-x 17 root   root   4096  2월 10  2017 ..
-r--r--r--  1 root   root      8  3월 24  2021 .baro_acl
-r--r--r--  1 root   root    305  7월  2 14:41 .baro_auth
-r--r--r--  1 root   root    290  6월 30 12:55 .baro_curl
-rwxr-xr-x  1 root   root  69149  4월  6 19:12 baro_auth
-rwxr-xr-x  1 root   root  65072  6월 29 16:36 baro_curl
drwxr-xr-x  2 root   root   4096  7월 20  2021 jilee
-rwxr-xr-x  1 root   root 152649  6월  9 08:19 pam_baro_auth.so
-rwxr-xr-x  1 root   root 116158  6월 30 12:54 pam_baro_curl.so
-rw-r--r--  1 root   root    150  6월 29 16:29 setcurl.sh
-rw-r--r--  1 root   root    221  6월 27 15:59 setenv.sh
```

3.3 Create BaroPAM configuration file

1) PAM authentication (.baro_auth)

The BaroPAM environment setting file must be created by executing the **baro_auth** program, and it must be located under **/usr/baropam**, the directory of the BaroPAM authentication module.

Format)

```
baro_auth -r rate_limit -R rate_time -t cycle_time -k key_method -e encrypt_flag -H hostname -A acl_type -a acl_filename -S secure_key -s filename
```

The configuration options of the BaroPAM configuration file are as follows.

Optino	Decumentation	Set value	Etc
-r	OTA key limited number of times (1~10)	3	
-R	OTA key time limit (15~600 sec)	30	
-t	OTA key authentication cycle (3~60 sec)	30	
-k	OTA key authentication method (app1, app256, app384, app512)	app512	
-e	Encryption of configuration files (yes or no)	no	
-H	Server's hostname (uname -n)	nurit.co.kr	
-A	Choose whether to allow or deny 2nd authentication	deny	
-a	ACL file name for the account to allow or deny from 2nd authentication (**file access permission is 444**)	/usr/baropam/.baro_acl	
-S	Secure key (license key) provided by the vendor	jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/	
-s	File name including the directory in which to create the BaroPAM configuration file	/usr/baropam/.baro_auth	

Note) The filename of the -s option is the name of the file including the directory where the BaroPAM configuration file will be created (**file access permission is 444**). If the hostname of the set server does not match, BaroPAM may not operate normally. If the hostname is changed, it must be reflected in the relevant item of the environment setting.

Ex of use)

```
[root] /usr/baropam > ./baro_auth -r 3 -R 30 -t 30 -k app512 -e no -H nurit.co.kr -A deny -a
/usr/baropam/.baro_acl -S jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/ -s /usr/baropam/.baro_auth
```

If the BaroPAM environment setting file is set for each account, connect to the account and proceed with the work. (Not root)

```
[root] /usr/baropam > ./baro_auth -r 3 -R 30 -t 30 -k app512 -e no -H nurit.co.kr -A deny -a
~/.baro_acl -S jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/ -s ~/.baro_auth
```

1) Your emergency one-time authentication keys are:
   The emergency OTA key is a super authentication key that can be used to access the SSH server again in case you lose it when the OTA key generator, the BaroPAM app, is unavailable, so it is good to write it down somewhere.

2) Enter "y" for all the questions that follow.
   Do you want me to update your "/usr/baropam/.baro_auth" file (y/n) y
   Preventing man-in-the-middle attacks (y/n) y

The contents set in .baro_auth, the BaroPAM environment setting file, are as follows.

```
[root] /usr/baropam > cat .baro_auth
" AUTH_KEY
" RATE_LIMIT 3 30
" KEY_METHOD app512
" CYCLE_TIME 30
" SECURE_KEY jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/
" ACL_NAME /usr/baropam/.baro_acl
" ACL_TYPE deny
" HOSTNAME nurit.co.kr
" DISALLOW_REUSE
33458936
19035576
15364353
54649370
84342192
```

The setting items of .baro_auth, a BaroPAM configuration file, are as follows.

Item	Decumentation	Set value	Etc
AUTH_KEY	Authentication delimiter (fixed)		
RATE_LIMIT	OTA key limit count (1~10), time limit (15~600 sec)	3 30	
KEY_METHOD	OTA key authentication method (app1, app256, app384, app512: app	app512	
CYCLE_TIME	OTA key authentication cycle (3~60 sec)	30	
SECURE_KEY	Secure key (license key) provided by the vendor	jlqlcHbVqdpj7b4PzBpM2DileBvmHFV/	
HOSTNAME	Server's hostname (uname -n)	nurit.co.kr	
ACL_TYPE	Differentiate between allow and deny in 2nd authentication	deny	
ACL_NAME	ACL Filename for the account to be allowed or excluded from 2nd	/usr/baropam/.baro_acl	

	authentication (file access permission is 444)		
DISALLOW_REUSE or ALLOW_REUSE	To prevent a man-in-the-middle attack, if "DISALLOW_REUSE" is set, other users cannot log in during the authentication cycle of the OTA key. If allowed, set "ALLOW_REUSE".	DISALLOW_REUSE	

2) cURL authentication (.baro_curl)

The name curl stands for "client URL" and was first released in 1997. That is, the client requests data from the server as a script. BaroPAM requests authentication by calling the http/https authentication site with curl.

The BaroPAM environment setting file must be created by executing the baro_curl program, and it must be located under /usr/baropam, the directory of the BaroPAM authentication module.

Format)

```
baro_curl -r rate_limit -R rate_time -t cycle_time -k key_method -e encrypt_flag -H hostname -u auth_url -s filename
```

The configuration options of the BaroPAM configuration file are as follows.

Option	Decumentation	Set value	Etc
-r	OTA key limited number of times (1~10)	3	
-R	OTA key time limit (15~600 sec)	30	
-t	OTA key authentication cycle (3~60 sec)	30	
-k	OTA key authentication method (app1, app256, app384, app512: app)	app512	
-e	Encryption of configuration files (yes or no)	no	
-H	Server's hostname (uname -n)	nurit.co.kr	
-u	The URL to be called includes parameters such as host name (hostname), user account (username), authentication cycle (cycle_time), and OTA key (auth_key)	http://1.23.456.789/baropam/web/result_curl.jsp	
-s	File name including the directory in which to create the BaroPAM configuration file	/usr/baropam/.baro_curl	

Note) The filename of the -s option is the name of the file including the directory where the BaroPAM configuration file will be created (file access permission is 444). If the hostname of the set server does not match, BaroPAM may not operate normally. If the hostname is changed, it must be reflected in the relevant item of the environment setting.

Ex of use)

```
[root] /usr/baropam > ./baro_curl -r 3 -R 30 -t 30 -k app512 -e no -H nurit.co.kr -u http://1.23.456.789/baropam/web/result_curl.jsp -s /usr/baropam/.baro_curl
```

1) Enter "y" for all the questions that follow.

```
Do you want me to update your "/usr/baropam/.baro_curl" file (y/n) y
Preventing man-in-the-middle attacks (y/n) y
```

The contents set in .baro_curl, a BaroPAM environment setting file, are as follows.

```
[root] /usr/baropam > cat .baro_curl
" AUTH_KEY
" RATE_LIMIT 3 30
" AUTH_URL http://1.23.456.789/baropam/web/result_curl.jsp
" KEY_METHOD app512
" CYCLE_TIME 30
" HOSTNAME baropam
" DISALLOW_REUSE
```

The setting items of **.baro_curl**, a BaroPAM configuration file, are as follows.

Item	Documentation	Set value	Etc
AUTH_KEY	Authentication delimiter (fixed)		
RATE_LIMIT	OTA key limit count (1~10), time limit (15~600 sec)	3 30	
AUTH_URL	The URL to be called includes parameters such as host name (hostname), user account (username), authentication cycle (cycle_time), and OTA key (auth_key)	http://1.23.456.789/baropam/web/result_curl.jsp	
KEY_METHOD	OTA key authentication method (app1, app256, app384, app512: app)	app512	
CYCLE_TIME	OTA key authentication cycle (3~60 sec)	30	
HOSTNAME	Server's hostname (uname -n)	nurit.co.kr	
DISALLOW_REUSE or ALLOW_REUSE	To prevent a man-in-the-middle attack, if "**DISALLOW_REUSE**" is set, other users cannot log in during the authentication cycle of the OTA key. If allowed, set "**ALLOW_REUSE**".	DISALLOW_REUSE	

3.4 BaroPAM environment settings

1) PAM authentication

To configure the BaroPAM module, enter it at the top as follows to configure **radiusd** files.

```
[root] /usr/baropam > vi /etc/pam.d/radiusd
auth       required     /usr/baropam/pam_baro_auth.so forward_pass secret=/usr/baropam/.baro_auth
encrypt=no
```

For reference, the **secret** parameter sets the name of the BaroPAM configuration file, and the **encrypt** parameter sets the encryption/decryption flag (**yes** or **no**) of the BaroPAM configuration file.

If the BaroPAM environment setting file is set for each account, the way to set the **radiusd** file to set the BaroPAM module is entered at the top as follows.

```
[root] /usr/baropam > vi /etc/pam.d/radiusd
#%PAM-1.0
auth          required       /usr/baropam/pam_baro_auth.so forward_pass secret=${HOME}/.baro_auth
encrypt=no
```

If you want to set different BaroPAM environment configuration files for each account in a specific directory instead of setting BaroPAM environment configuration files for each account, enter the following at the top to configure the BaroPAM module in the sshd file.

```
[root] /usr/baropam > vi /etc/pam.d/radiusd
#%PAM-1.0
auth                    required                /usr/baropam/pam_baro_auth.so    forward_pass
secret=/usr/baropam/radius/.${USER}_auth encrypt=no
```

For programs like filezilla that cannot perform "Interactive process", the only way is to use the forward_pass option in PAM to enter the OTA key when entering the password. In this case, the openssh client, RDP (Remote Desktop Protocol) of Windows, Radius, filezilla, etc. all have no choice but to input like this.

When entering the OTA key like a password in the password input window (Password:) using forward_pass, enter the password first and then enter the OTA key without spaces. For example, if the password is "baropam" and the OTA key is "123456", enter "baropam123456".

Using forward_pass, you can enable 2nd authentication for most services that require authentication.

2) cURL authentication

To configure the BaroPAM module, enter it at the top as follows to configure radiusd files.

```
[root] /usr/baropam > vi /etc/pam.d/radiusd
auth       required     /usr/baropam/pam_baro_curl.so forward_pass secret=/usr/baropam/.baro_curl
encrypt=no
```

For reference, the secret parameter sets the name of the BaroPAM configuration file, and the encrypt parameter sets the encryption/decryption flag (yes or no) of the BaroPAM configuration file.

For programs like filezilla, which cannot perform "Interactive process", the only way is to use the forward_pass option in PAM to enter the password and OTA key together when entering the password. In this case, the openssh client, RDP (Remote Desktop Protocol) of Windows, Radius, filezilla, etc. all have no choice but to input like this.

When entering the OTA key like a password in the password input window (**Password:**) using **forward_pass**, enter the password first and then enter the OTA key without spaces. For example, if the password is "**baropam**" and the OTA key is "**123456**", enter "**baropam123456**".

Using **forward_pass**, you can enable **2nd authentication** for most services that require authentication.

3) ACL(Access Control list) 설정

1) In the case of PAM authentication When using the BaroPAM module, if you need to exclude from the ACL for the account to be excluded from secondary authentication, create an ACL file in the directory set when setting up the BaroPAM environment, and enter the account to be excluded as follows. (**The file access permission for .baro_acl must be set to 444.**)

```
[root] /usr/baropam > vi .baro_acl
barokey
baropam
```

4) NTP(Network Time Protocol) settings

Since BaroPAM is a time synchronization method, if the server's time is different from the current time, login to the server may not be possible because the OTA keys do not match.

Recently, as a method of time synchronization (time server time synchronization) for information assets, the system time can be set to the current time in the root account using NTP (Network Time Protocol).

To use NTP, the NTP package must be installed by default. To check the installation, run the following command. If it is not installed, use the command "**yum install ntp**" for Redhat, CentOS 8 or lower, and "**sudo apt-get install ntp**" for others.

```
[root]# rpm -qa | grep ntp
ntp-4.2.2p1-18.el5.centos
chkfontpath-1.10.1-1.1
```

The following command can be used to register the ntpd service in the startup program when booting the server and to check whether ntp is activated.

```
[root]# chkconfig ntpd on
[root]# chkconfig —list | grep ntp
ntpd            0:off  1:off  2:on  3:on  4:on  5:on  6:off
```

Check whether the ntpd daemon is active when booting the server using chkconfig. If it is off in level 3 and 5, it is not activated automatically. To activate automatically, you must change 3 and 5 to on (active) with the following command.

```
[root]# chkconfig —level 3 ntpd on
[root]# chkconfig —level 5 ntpd on
```

NTP servers operating in Korea are as follows.

```
server kr.pool.ntp.org
server time.bora.net
```

Set the NTP server operating in Korea in "**/etc/ntp.conf**", the configuration file for the ntpd daemon configuration, as follows.

```
[root]# vi /etc/ntp.conf
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#server 0.centos.pool.ntp.org
#server 1.centos.pool.ntp.org
#server 2.centos.pool.ntp.org
#server 3.centos.pool.ntp.org
server kr.pool.ntp.org iburst
server time.bora.net   iburst
```

The **iburst** option is a kind of option setting that shortens the time required for synchronization.

After the setup for the ntpd daemon setup is complete, it is absolutely necessary to restart the NTP daemon after confirming that the NTP setup has been properly added.

```
[root]# /etc/init.d/ntpd restart
Stopping ntpd: [  OK  ]
Starting ntpd: [  OK  ]
```

You can check the ntpd time with the following command.

```
[root]# ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*121.174.142.82  220.73.142.66    3 u  791 1024  377    9.333   -4.250   0.428
+time.bora.net   58.224.35.2      3 u  654 1024  367    2.926  -27.295  24.481
 183.110.225.61  .INIT.          16 u    - 1024    0    0.000    0.000   0.000
 LOCAL(0)        .LOCL.          10 l   39   64  377    0.000    0.000   0.001
```

* The displayed ip is the ntp server getting the current time

To use NTP, the NTP package must be installed by default. To check the installation, run the following command. If it is not installed, use the "**dnf install chrony**" command to install Redhat, CentOS 8 or later versions.

```
[root@baropam ~]# rpm -qa | grep chrony
chrony-3.5-1.el8.x86_64
```

NTP servers operating in Korea are as follows.

```
server kr.pool.ntp.org
server time.bora.net
```

Set the NTP server operating in Korea in "**/etc/chrony.conf**", the configuration file for the ntpd daemon configuration, as follows.

```
[root@baropam ~]# vi /etc/chrony.conf

# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#pool 2.centos.pool.ntp.org iburst
server kr.pool.ntp.org iburst
server time.bora.net   iburst

# Record the rate at which the system clock gains/losses time.
driftfile /var/lib/chrony/drift

# Allow the system clock to be stepped in the first three updates
# if its offset is larger than 1 second.
makestep 1.0 3

# Enable kernel synchronization of the real-time clock (RTC).
rtcsync

# Enable hardware timestamping on all interfaces that support it.
#hwtimestamp *

# Increase the minimum number of selectable sources required to adjust
# the system clock.
#minsources 2

# Allow NTP client access from local network.
allow 192.168.0.0/16

# Serve time even if not synchronized to a time source.
#local stratum 10

# Specify file containing keys for NTP authentication.
keyfile /etc/chrony.keys

# Get TAI-UTC offset and leap seconds from the system tz database.
leapsectz right/UTC

# Specify directory for log files.
logdir /var/log/chrony

# Select which information is logged.
#log measurements statistics tracking
```

After the setup for the ntpd daemon setup is complete, it is absolutely necessary to restart the NTP daemon after confirming that the NTP setup has been properly added. (Starting chrony service and registering drive when booting)

```
[root@baropam ~]# sudo systemctl enable chronyd
[root@baropam ~]# sudo systemctl restart chronyd
```

You can check the ntpd time with the following command.

List of servers receiving time / list of servers registered in chrony.conf file)

```
[root@baropam ~]# chronyc sources
210 Number of sources = 2
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* ec2-54-180-134-81.ap-nor>     2   6   377    43   -349us[-1059us] +/-   24ms
^- time.bora.net                 2   6   377    42  +1398us[+1398us] +/-   90ms
```

Server information receiving time)

```
[root@baropam ~]# chronyc tracking
Reference ID    : 36B48651 (ec2-54-180-134-81.ap-northeast-2.compute.amazonaw)
Stratum         : 3
Ref time (UTC)  : Sun Mar 22 07:07:43 2020
System time     : 0.000130027 seconds slow of NTP time
Last offset     : -0.000710122 seconds
RMS offset      : 0.000583203 seconds
Frequency       : 19.980 ppm fast
Residual freq   : +0.142 ppm
Skew            : 3.235 ppm
Root delay      : 0.013462566 seconds
Root dispersion : 0.017946836 seconds
Update interval : 65.0 seconds
Leap status     : Normal
```

Check information such as time status and synchronization)

```
[root@baropam ~]# timedatectl status
               Local time: Sun 2020-03-22 16:08:45 KST
           Universal time: Sun 2020-03-22 07:08:45 UTC
                 RTC time: Sun 2020-03-22 07:08:44
                Time zone: Asia/Seoul (KST, +0900)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

4. Install and configure MySQL/MariaDB

4.1 Install MariaDB

```
[root@localhost ~]# dnf -y install mariadb-server
```

When the MaraDB installation process is completed normally, start MariaDB using the following command.

```
[root@localhost ~]# systemctl start mariadb
```

You need to take a few steps to improve MariaDB security options by running the scripts provided with MariaDB.

```
[root@localhost ~]# mysql_secure_installation
```

A series of prompts will appear, and if you don't know you set a password, press Enter when prompted.

```
Enter current password for root (enter for none): Enter
```

Next, confirm that you want to set a new **root** password and set a strong password.

```
Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.
Set root password? [Y/n] y
New password: baropam
Re-enter new password: baropam
Password updated successfully!
Reloading privilege tables..
... Success!
```

All you have to do is press Enter for the prompt that follows.

Remove anonymous users.

```
Remove anonymous users? [Y/n] y
... Success!
```

Do not allow **root** login remotely.

```
Disallow root login remotely? [Y/n] y
... Success!
```

Remove the test database.

```
Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
```

```
- Removing privileges on test database...
... Success!
```

Reload the privilege table.

```
Reload privilege tables now? [Y/n] y
... Success!
Cleaning up...
```

4.2 MariaDB configuration

First create a database and database user for FreeRADIUS, then create a database and a user identified by a password.

Ex)

```
Database: radius
User: radius
Password: baropam
```

You can change the user and password to whatever you want, but you'll need to pay attention to the configuration you'll do later to change the values appropriately.

root로 MySQL/MariaDB 콘솔에 액세스하여 시작한다.

```
[root@baropam /]# mysql -uroot -p
Enter password: baropam
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.26 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Execute command to create database and user.

```
MariaDB [(none)]> CREATE DATABASE radius;
MariaDB [(none)]> GRANT ALL ON radius.* TO radius@localhost IDENTIFIED BY "baropam";
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> use radius
```

Next, create the RADIUS MySQL schema as the newly created database.

```
##########################################################################
# $Id: 2d0fb7e137f9d6f6a2a48d65e013841ed2bfadf9 $                        #
#                                                                        #
```

```
#  schema.sql                        rlm_sql - FreeRADIUS SQL Module      #
#                                                                         #
#     Database schema for MySQL rlm_sql module                             #
#                                                                         #
#     To load:                                                             #
#         mysql -uroot -prootpass radius < schema.sql                      #
#                                                                         #
#                                    Mike Machado <mike@innercite.com>    #
###########################################################################
#
# Table structure for table 'radacct'
#
CREATE TABLE IF NOT EXISTS radacct (
  radacctid bigint(21) NOT NULL auto_increment,
  acctsessionid varchar(64) NOT NULL default '',
  acctuniqueid varchar(32) NOT NULL default '',
  username varchar(64) NOT NULL default '',
  realm varchar(64) default '',
  nasipaddress varchar(15) NOT NULL default '',
  nasportid varchar(15) default NULL,
  nasporttype varchar(32) default NULL,
  acctstarttime datetime NULL default NULL,
  acctupdatetime datetime NULL default NULL,
  acctstoptime datetime NULL default NULL,
  acctinterval int(12) default NULL,
  acctsessiontime int(12) unsigned default NULL,
  acctauthentic varchar(32) default NULL,
  connectinfo_start varchar(50) default NULL,
  connectinfo_stop varchar(50) default NULL,
  acctinputoctets bigint(20) default NULL,
  acctoutputoctets bigint(20) default NULL,
  calledstationid varchar(50) NOT NULL default '',
  callingstationid varchar(50) NOT NULL default '',
  acctterminatecause varchar(32) NOT NULL default '',
  servicetype varchar(32) default NULL,
  framedprotocol varchar(32) default NULL,
  framedipaddress varchar(15) NOT NULL default '',
  framedipv6address varchar(45) NOT NULL default '',
  framedipv6prefix varchar(45) NOT NULL default '',
  framedinterfaceid varchar(44) NOT NULL default '',
  delegatedipv6prefix varchar(45) NOT NULL default '',
  PRIMARY KEY (radacctid),
  UNIQUE KEY acctuniqueid (acctuniqueid),
  KEY username (username),
  KEY framedipaddress (framedipaddress),
  KEY framedipv6address (framedipv6address),
  KEY framedipv6prefix (framedipv6prefix),
  KEY framedinterfaceid (framedinterfaceid),
  KEY delegatedipv6prefix (delegatedipv6prefix),
  KEY acctsessionid (acctsessionid),
  KEY acctsessiontime (acctsessiontime),
  KEY acctstarttime (acctstarttime),
  KEY acctinterval (acctinterval),
```

```
  KEY acctstoptime (acctstoptime),
  KEY nasipaddress (nasipaddress)
) ENGINE = INNODB;

#
# Table structure for table 'radcheck'
#
CREATE TABLE IF NOT EXISTS radcheck (
  id int(11) unsigned NOT NULL auto_increment,
  username varchar(64) NOT NULL default '',
  phone_no varchar(32) NOT NULL default '',
  cycle_time varchar(6) NOT NULL default '60',
  attribute varchar(64)  NOT NULL default '',
  op char(2) NOT NULL DEFAULT '==',
  value varchar(253) NOT NULL default '',
  PRIMARY KEY  (id),
  KEY username (username(32))
);

#
# Table structure for table 'radgroupcheck'
#
CREATE TABLE IF NOT EXISTS radgroupcheck (
  id int(11) unsigned NOT NULL auto_increment,
  groupname varchar(64) NOT NULL default '',
  attribute varchar(64)  NOT NULL default '',
  op char(2) NOT NULL DEFAULT '==',
  value varchar(253)  NOT NULL default '',
  PRIMARY KEY  (id),
  KEY groupname (groupname(32))
);

#
# Table structure for table 'radgroupreply'
#
CREATE TABLE IF NOT EXISTS radgroupreply (
  id int(11) unsigned NOT NULL auto_increment,
  groupname varchar(64) NOT NULL default '',
  attribute varchar(64)  NOT NULL default '',
  op char(2) NOT NULL DEFAULT '=',
  value varchar(253)  NOT NULL default '',
  PRIMARY KEY  (id),
  KEY groupname (groupname(32))
);

#
# Table structure for table 'radreply'
#
CREATE TABLE IF NOT EXISTS radreply (
  id int(11) unsigned NOT NULL auto_increment,
  username varchar(64) NOT NULL default '',
  attribute varchar(64) NOT NULL default '',
  op char(2) NOT NULL DEFAULT '=',
```

```
  value varchar(253) NOT NULL default '',
  PRIMARY KEY  (id),
  KEY username (username(32))
);

#
# Table structure for table 'radusergroup'
#
CREATE TABLE IF NOT EXISTS radusergroup (
  id int(11) unsigned NOT NULL auto_increment,
  username varchar(64) NOT NULL default '',
  groupname varchar(64) NOT NULL default '',
  priority int(11) NOT NULL default '1',
  PRIMARY KEY  (id),
  KEY username (username(32))
);

#
# Table structure for table 'radpostauth'
#
CREATE TABLE IF NOT EXISTS radpostauth (
  id int(11) NOT NULL auto_increment,
  username varchar(64) NOT NULL default '',
  pass varchar(64) NOT NULL default '',
  reply varchar(32) NOT NULL default '',
  authdate timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY  (id),
  KEY username (username(32))
) ENGINE = INNODB;

#
# Table structure for table 'nas'
#
CREATE TABLE IF NOT EXISTS nas (
  id int(10) NOT NULL auto_increment,
  nasname varchar(128) NOT NULL,
  shortname varchar(32),
  type varchar(30) DEFAULT 'other',
  ports int(5),
  secret varchar(60) DEFAULT 'secret' NOT NULL,
  server varchar(64),
  community varchar(50),
  description varchar(200) DEFAULT 'RADIUS Client',
  PRIMARY KEY (id),
  KEY nasname (nasname)
);
```

5. FreeRADIUS integration test

5.1 PAM, cURL authentication

1) Create an account to test

Create a local account to test with.

```
[root@localhost ~]# useradd raduser
[root@localhost ~]# passwd raduser
Changing password for user raduser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

To execute a shell script related to the user's login information (login-ID, phone number), go to the BaroPAM registration directory (**/usr/baropam/radius**) and execute it.

Note) When executing a shell script, arguments or argument values must be separated by spaces.

Ex) Shell script to create a user (login-id) (if auths=pam) - setuser.sh

```
#!/bin/sh

export LANG=C
ENV_HOME=/usr/baropam/radius;
ACC_HOME=/home/$1

userdel -rf $1
Wrm ${ENV_HOME}/.$1_auth

useradd -d ${ACC_HOME} -m -s /bin/bash $1
echo $2 | passwd -stdin $1

Wcp ${ENV_HOME}/.baro_auth ${ENV_HOME}/.$1_auth

sed -i "s/01012341234/$3/g" ${ENV_HOME}/.$1_auth
```

When running the shell script (setuser.sh) that creates the user(login-ID), the parameters are as follows.
$1 : Login-ID to create
$2 : Login-ID's password
$3 : Login-ID phone number

```
[root@localhost ~]# sh setuser.sh nurit baropam 01027714076
```

Ex) Shell script to change the password of a user(login-id) (if auths=pam) - setpasswd.sh

```
#!/bin/sh
```

```
export LANG=C

echo $2 | passwd -stdin $1
```

When executing the shell script (setpasswd.sh) to change the user (login-ID) password, the parameters are as follows.
$1 : Login-ID
$2 : Change password

```
[root@localhost ~]# sh setpasswd.sh nurit !@Baropam#
```

Ex) Shell script to change the phone number of a user(login-ID) (if auths=pam) - setphone.sh

```
#!/bin/sh

export LANG=C
ENV_HOME=/usr/baropam/radius;

sed -i "s/$2/$3/g" ${ENV_HOME}/.$1_auth
```

When executing the shell script (setphone.sh) that changes the phone number of the user(login-ID), the parameters are as follows.
$1 : Login-ID
$2 : Phone number before change
$3 : Phone number after change

```
[root@localhost ~]# sh setphone.sh nurit 01012341234 01027714076
```

Ex) Shell script that changes the password and phone number of the user(login-ID) (if auths=pam) - chgpasswd.sh

```
#!/bin/sh

export LANG=C

echo $2 | passwd -stdin $1

sed -i "s/$3/$4/g" ${ENV_HOME}/.$1_auth
```

When executing the shell script (setpasswd.sh) that changes the user (login-ID) password and phone number, the parameters are as follows.
$1 : Login-ID
$2 : Change password
$3 : Phone number before change
$4 : Phone number after change

```
[root@localhost ~]# sh chgpasswd.sh nurit !@Baropam# 01012341234 01027714076
```

Ex) Shell script to delete user(login-id) (if auths=pam) - deluser.sh

```
#!/bin/sh
```

```
export LANG=C
ENV_HOME=/usr/baropam/radius;
ACC_HOME=/home/$1

userdel -rf $1
Wrm ${ENV_HOME}/.$1_auth
```

When executing the shell script (deluser.sh) that deletes a user(login-ID), the parameters are as follows.
$1 : Login-ID to delete

```
[root@localhost ~]# sh deluser.sh nurit
```

Ex) Shell script to generate a user(login-id) (if auth=http) - setuser.sh

```
#!/bin/sh

export LANG=C
ENV_HOME=/usr/baropam/radius;
ACC_HOME=/home/$1

userdel -rf $1
Wrm ${ENV_HOME}/.$1_auth

useradd -d ${ACC_HOME} -m -s /bin/bash $1
echo $1 | passwd -stdin $1

Wcp ${ENV_HOME}/.baro_auth ${ENV_HOME}/.$1_auth
```

When running the shell script (setuser.sh) that creates the user(login-ID), the parameters are as follows.
$1 : Login-ID to create

```
[root@localhost ~]# sh setuser.sh nurit baropam
```

Ex) Shell script to delete user(login-id) (if auths=http) - deluser.sh

```
#!/bin/sh

export LANG=C
ENV_HOME=/usr/baropam/radius;
ACC_HOME=/home/$1

userdel -rf $1
Wrm ${ENV_HOME}/.$1_auth
```

When executing the shell script (deluser.sh) that deletes a user(login-ID), the parameters are as follows.
$1 : Login-ID to delete

```
[root@localhost ~]# sh deluser.sh nurit
```

2) Running in FreeRADIUS debug mode

Open a new ssh session and run **radiusd** in debug mode as follows.

```
[root@localhost ~]# radiusd -X
```

If you get the following error when running **radiusd** in debug mode:

```
Then run radiusd -X but I have this error:

Failed reading private key file /etc/raddb/certs/server.pem:error:06065064:digital envelope
routines:EVP_DecryptFinal_ex:bad decrypt
rlm_eap_tls: Failed initializing SSL context
rlm_eap (EAP): Failed to initialise rlm_eap_tls
/etc/raddb/mods-enabled/eap[17]: Instantiation failed for module "eap"
```

After doing the following, run **radiusd** as follows.

```
[root@localhost ~]# cd  /etc/raddb/certs
[root@localhost ~]# ./bootstrap
```

First, switch to the ssh session and test.

Format)

```
radtest <username> (<pasword><verification code>) localhost 0 baropam
```

Run the FreeRADIUS test with BaroPAM as follows.

```
[root@localhost ~]# radtest root baropam024747 localhost 0 baropam
Sent Access-Request Id 243 from 0.0.0.0:39668 to 127.0.0.1:1812 length 74
        User-Name = "root"
        User-Password = "baropam903481"
        NAS-IP-Address = 10.0.2.15
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "baropam903481"
Received Access-Accept Id 243 from 127.0.0.1:1812 to 127.0.0.1:39668 length 20
```

3) Running the FreeRADIUS daemon

Run **radiusd**, the FreeRADIUS daemon, in the background as follows.

```
[root@localhost ~]# radiusd -s &
[1] 1961
```

Check if **radiusd**, the FreeRADIUS daemon, is normally started as follows.

```
[root@localhost ~]# ps -ef|grep radiusd
root        1961    1818  0 14:11 pts/1    00:00:00 radiusd -s
root        1964    1818  0 14:11 pts/1    00:00:00 grep --color=auto radiusd
```

Check the UDP port **1812** used by the FreeRADIUS daemon **radiusd** as follows.

```
[root@localhost ~]# netstat -an | grep 1812
udp        0      0 127.0.0.1:18120         0.0.0.0:*
udp        0      0 0.0.0.0:1812            0.0.0.0:*
udp6       0      0 :::1812                 :::*
```

Check the logs authenticated by BaroPAM by linking FreeRADIUS as follows.

```
[root@localhost ~]# tail -f /var/log/secure
Mar 26 13:54:11 localhost radiusd(pam_baro_auth)[1857]: Try to update RATE_LIMIT line.[3 30
1616734451]
Mar 26 13:56:46 localhost radiusd(pam_baro_auth)[1857]: Try to update RATE_LIMIT line.[3 30
1616734606]
Mar 26 14:00:48 localhost radiusd(pam_baro_auth)[1934]: Try to update RATE_LIMIT line.[3 30
1616734848]
Mar 26 14:00:48 localhost radiusd(pam_baro_auth)[1934]: Invalid verification code
Mar 26 14:00:48 localhost radiusd[1934]: pam_unix(radiusd:auth): authentication failure;
logname=root uid=0 euid=0 tty= ruser= rhost=  user=scjoo1
Mar 26 14:01:13 localhost radiusd(pam_baro_auth)[1934]: Try to update RATE_LIMIT line.[3 30
1616734873]
Mar 26 14:01:36 localhost radiusd(pam_baro_auth)[1934]: Try to update RATE_LIMIT line.[3 30
1616734873 1616734896]
Mar 26 14:02:15 localhost radiusd(pam_baro_auth)[1934]: Try to update RATE_LIMIT line.[3 30
1616734935]
```

4) Shutdown the FreeRADIUS daemon

First, check if **radiusd**, the FreeRADIUS daemon, is running as follows.

```
[root@localhost ~]# ps -ef|grep radiusd
root       1961    1818  0 14:11 pts/1    00:00:00 radiusd -s
root       1964    1818  0 14:11 pts/1    00:00:00 grep --color=auto radiusd
```

After checking the process ID (**1961**) of **radiusd**, the FreeRADIUS daemon, terminate it as follows.

```
[root@localhost ~]# kill -9 1961
```

Reference) FreeRADIUS related log file: /var/log/radius/radius.log

5.2 SQL authentication

1) Work ahead

MariaDB's UDF (User Defined Function) is used in MariaDB to call an external program written in C or C++ or send data.

It must be written in C or C++, and the operating system must support dynamic loading.

Copy the attached file (**libbaroudf.so**) to the directory with the result of the following command

in the mysql shell.

```
mysql> SHOW VARIABLES LIKE 'plugin_dir';
+---------------+-------------------------+
| Variable_name | Value                   |
+---------------+-------------------------+
| plugin_dir    | /usr/lib64/mysql/plugin/ |
+---------------+-------------------------+
1 row in set (0.00 sec)
```

① Copy the BaroPAM module

```
[root@localhost ~]# cp libbaroudf.so /usr/lib64/mysql/plugin/
```

② Create BaroPAM function

```
mysql> create function TO_VERIFYKEYL        returns string soname 'libbaroudf.so';
Query OK, 0 rows affected (0.01 sec)
```

Note) TO_VERIFYKEYL function

- NAME
  TO_VERIFYKEYL

- SYNOPSIS
  void * TO_VERIFYKEYL(const void *username, const void *phone_no, const void *cycle_time)

- DESCRIPTION
  OTA key authentication function
  username: Sets the ID entered in the login-ID item of the login screen.
  phone_no: Set the smartphone number for each user only with numbers.
  cycle_time: Sets the OTA key generation cycle (3~60 sec) specified for each user.

- RETURN VALUES
  Returns the generated OTA key

Note) If the OTA key generation cycle specified for each user and the smartphone number is different from the OTA key generator, verification may fail because the OTA key is different. The information must match.

③ Change authorize_check_query

In /etc/raddb/mods-config/sql/main/mysql/**queries.conf** file, **change authorize_check_query to TO_VERIFYKEYL function.**

Ex) In case of validating only OTA key

```
authorize_check_query = "W
        SELECT id, username, attribute, TO_VERIFYKEYL(username, phone_no, cycle_time) as value, op
W
        FROM ${authcheck_table} W
        WHERE username = '%{SQL-User-Name}' W
        ORDER BY id"
```

Ex) When verifying (password + OTA key) in a combined form

```
authorize_check_query = "W
        SELECT  id,  username,  attribute,  CONCAT(value,  TO_VERIFYKEYL(username,  phone_no,
cycle_time)) as value, op W
        FROM ${authcheck_table} W
        WHERE username = '%{SQL-User-Name}' W
        ORDER BY id"
```

④ Create user to test

To test, add the following values to the table.

```
MariaDB [(none)]> insert into radgroupreply (groupname,attribute,op,value) values ('baropam','Auth-
Type',':=', 'Local');

MariaDB [(none)]> insert into radcheck (username ,phone_no,cycle_time,attribute,op,value) values
('admin1','01027714076','60','Cleartext-Password',':=','admin1');

MariaDB [(none)]> insert into radusergroup (username,groupname) values ('admin1', 'baropam');
```

2) Running in FreeRADIUS debug mode

Open a new ssh session and run **radiusd** in debug mode as follows.

```
[root@localhost ~]# radiusd -X
```

If you get the following error when running **radiusd** in debug mode:

```
Then run radiusd -X but I have this error:

Failed reading private key file /etc/raddb/certs/server.pem:error:06065064:digital envelope
routines:EVP_DecryptFinal_ex:bad decrypt
rlm_eap_tls: Failed initializing SSL context
rlm_eap (EAP): Failed to initialise rlm_eap_tls
/etc/raddb/mods-enabled/eap[17]: Instantiation failed for module "eap"
```

After doing the following, run radiusd as follows.

```
[root@localhost ~]# cd  /etc/raddb/certs
[root@localhost ~]# ./bootstrap
```

First, switch to the ssh session and test.

Format)

```
radtest <username> (<pasword><verification code>) localhost 0 baropam

or

radtest <username> (<verification code>) localhost 0 baropam
```

Run the FreeRADIUS test with BaroPAM as follows

Ex) When verifying a OTA key

```
[root] /usr/baropam > radtest admin1 969533 localhost 0 baropam
Sent Access-Request Id 66 from 0.0.0.0:36066 to 127.0.0.1:1812 length 76
        User-Name = "admin1"
        User-Password = "969533"
        NAS-IP-Address = 10.0.2.15
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "969533"
Received Access-Accept Id 66 from 127.0.0.1:1812 to 127.0.0.1:36066 length 20
```

Ex) When verifying (password + OTA key) in a combined form

```
[root] /usr/baropam > radtest admin1 admin1969533 localhost 0 baropam
Sent Access-Request Id 66 from 0.0.0.0:36066 to 127.0.0.1:1812 length 76
        User-Name = "admin1"
        User-Password = "admin1969533"
        NAS-IP-Address = 10.0.2.15
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "admin1969533"
Received Access-Accept Id 66 from 127.0.0.1:1812 to 127.0.0.1:36066 length 20
```

3) Running the FreeRADIUS daemon

Run **radiusd**, the FreeRADIUS daemon, in the background as follows.

```
[root@localhost ~]# radiusd -s &
[1] 1961
```

Check if **radiusd**, the FreeRADIUS daemon, is normally started as follows.

```
[root@localhost ~]# ps -ef|grep radiusd
root        1961    1818  0 14:11 pts/1    00:00:00 radiusd -s
root        1964    1818  0 14:11 pts/1    00:00:00 grep --color=auto radiusd
```

Check the UDP port **1812** used by the FreeRADIUS daemon **radiusd** as follows.

```
[root@localhost ~]# netstat -an | grep 1812
udp        0      0 127.0.0.1:18120         0.0.0.0:*
udp        0      0 0.0.0.0:1812            0.0.0.0:*
udp6       0      0 :::1812                 :::*
```

4) Shutdown the FreeRADIUS daemon

First, check if **radiusd**, the FreeRADIUS daemon, is running as follows.

```
[root@localhost ~]# ps -ef|grep radiusd
root        1961    1818  0 14:11 pts/1    00:00:00 radiusd -s
root        1964    1818  0 14:11 pts/1    00:00:00 grep --color=auto radiusd
```

After checking the process ID (**1961**) of **radiusd**, the FreeRADIUS daemon, terminate it as follows.

```
[root@localhost ~]# kill -9 1961
```

Reference) FreeRADIUS related log file: /var/log/radius/radius.log

6. About BaroPAM

Version 1.0 - Official Release - 2016.12.1

http://www.nurit.co.kr

Company: Nurit Co., Ltd.
Registration Number: 258-87-00901
CEO: Jongil Lee
Tel: +8210-2771-4076(Technical support, sales inquiry)
email: mc529@nurit.co.kr
Address: #913, 15, Magokjungang 2-ro, Gangseo-gu, Seoul (Magok-dong, Magok Techno Tower 2)