SSLVPN Two - Factor Authentication with Google Authenticator

Hillstone Networks Inc.

1 Background

1.1 Two-Factor Authentication

Two-factor authentication (also known as 2FA) is a type (subset) of multi-factor authentication. It is a method of confirming a user's claimed identity by utilizing a combination of two different factors: 1) something they know, 2) something they have, or 3) something they are.

The good example is a user-controlled password with a one-time password (OTP) or code generated or received by an authenticator (e.g. a security token or smartphone) that only the user possesses.

1.2 Google Authenticator

Google Authenticator is a software-based authenticator that implements two-step verification services using the Time-based One-time Password Algorithm (TOTP; specified in RFC 6238) and HMAC-based One-time Password algorithm (HOTP; specified in RFC 4226), for authenticating users of mobile applications by Google.

When logging into a site supporting Authenticator (including Google services) or using Authenticator-supporting third-party applications such as password managers or file hosting services, Authenticator generates a six- to eight-digit one-time password which users must enter in addition to their usual login details.

2 FreeRADIUS & Google Authenticator Two-Factor Authentication

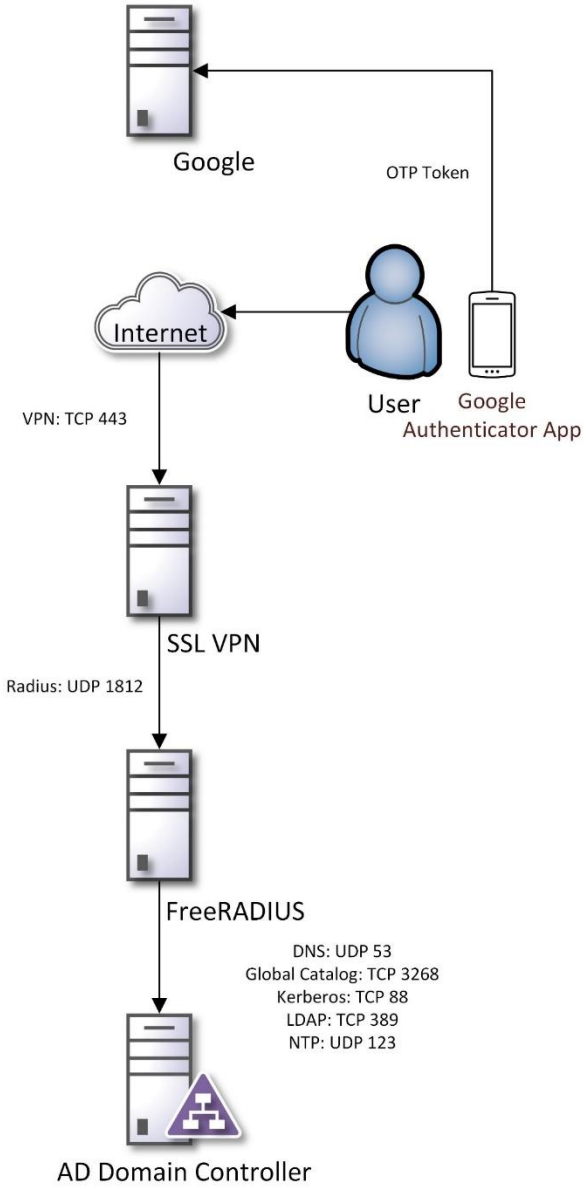
Google Authenticator is a great free dual factor authentication system. "The Google Authenticator project includes implementations of one-time passcode generators for several mobile platforms". It can be used in conjunction with FreeRADIUS to provide Free 2 factor authentication.

This all works because of a library called PAM. PAM is "Pluggable Authentication Modules" for Linux system user and password authentication. Google Authenticator has a PAM module that is included as part of the project. PAM is the glue that allows FreeRADIUS to talk to Google Authenticator.

FreeRADIUS is a popular open source radius server. Radius is a standardized authentication system that can be used to authenticate many different devices including VPNs, Routers, Switches, Computers, and much more.

The objective of this document is to provide a free two-factor authentication solution for use with VPN solutions.

Below is the architecture of this solution：



At first, you will need to complete a minimal installation of CentOS 7 build 1503 or RHEL 7.1 and yum update.

In addition, consistent and accurate time is a key requirement for the operation of this solution. The FreeRADIUS host will be utilizing SSSD integration with Active Directory and as such both must have the same time. In addition, Google Authenticator service and the device with the Google Authenticator App must have consistent time as well if using time-based One-Time Passwords (OTP). If problems occur during this tutorial with either SSSD

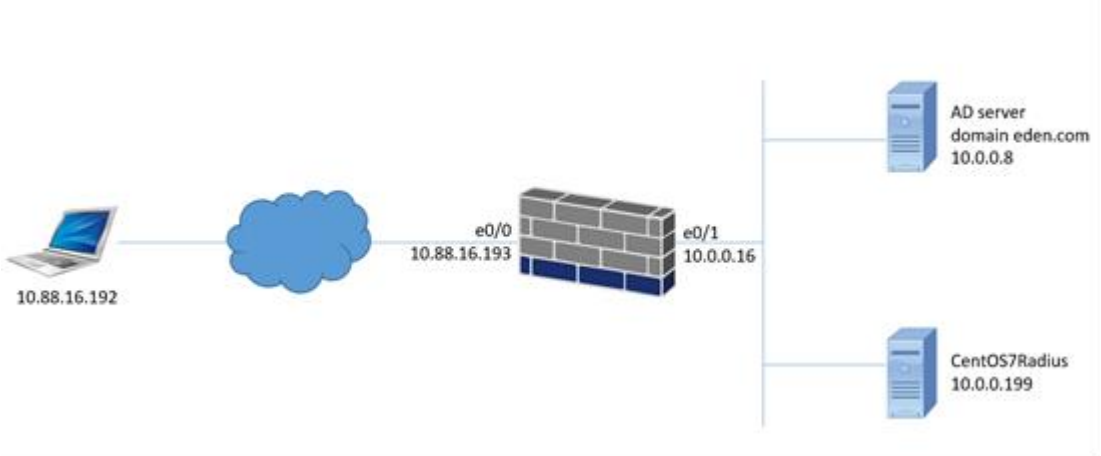
or Google Authenticator, verify the time is correct.

Required Components:

- CentOS 7 (1503) or Red Hat Enterprise Linux 7.1 Minimal
- FreeRADIUS
- System Security Services Daemon (SSSD)
- Google Authenticator Pam Library, Service, & APP
- Pluggable Authentication Module (PAM)

3 Hillstone SSLVPN 2FA Solution

Test Topology:



Configuration Steps:

1. Install CentOS 7

Check and set correct time

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

Set hostname

```
[root@localhost ~]# hostnamectl set-hostname CentOS7Radius
```

YUM update

```
[root@localhost ~]# yum update
```

Disable SELinux and firewall

```
[root@localhost ~]# systemctl stop firewalld.service
[root@localhost ~]# systemctl disable firewalld.service
[root@localhost ~]# vi /etc/selinux/config
SELINUX=disabled
[root@localhost ~]# reboot
```

2. Install and Configure FreeRADIUS

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

Change both user and group to root

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

Note:

This solution's use of FreeRADIUS must run as root to access the .google_authenticator in the user's home directory.

Edit sites-enabled/default

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

Uncomment pam

```
        #  Pluggable Authentication Modules.
        pam
```

Enable PAM

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

Configure clients.conf

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

Add firewall as a client

```
client 10.0.0.16{

ipaddr = 10.0.0.16
secret = hillstone
require_message_authenticator = no
nas_type = other

}
```

Change auth-type to PAM

```
# vi /etc/raddb/users
```

Find below

```
#DEFAULT Group == "disabled", Auth-Type := Reject
```

```
# Reply-Message = "Your account has been disabled."
#
```

Update to

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


DEFAULT Auth-Type := PAM
```

3. (Optional) Test FreeRADIUS local Unix account

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

Open a new ssh session and run radiusd in debug mode

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

Switch to first ssh session and test

```
[root@localhost ~]# radtest raduser your_password localhost 0 testing123
Sent Access-Request Id 83 from 0.0.0.0:51250 to 127.0.0.1:1812 length 77
        User-Name = "raduser"
        User-Password = "your_password"
        NAS-IP-Address = 10.0.0.199
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "your_password"
Received Access-Accept Id 83 from 127.0.0.1:1812 to 0.0.0.0:0 length 20
[root@localhost ~]#
```

Received Access-Accept should be the response, otherwise you will receive a reject. If so, backup and check your work and correct errors before proceeding.

4. Install and Configure SSSD

```
[root@localhost ~]# yum install sssd realmd adcli
[root@localhost ~]# yum install oddjob oddjob-mkhomedir sssd samba-common-tools
[root@localhost ~]# # realm join eden.com
Password for Administrator:
```

Note:

If you see error "realm: Couldn't connect to realm service: Error calling StartServiceByName for org.freedesktop.realmd: Timeout was reached", please reboot the system and try again.

Test SSSD

```
Last login: Wed Sep  4 17:21:13 2019
[adtom@eden.com@centos7radius ~]$
```

Test FreeRADIUS with a SSSD account

Run freeradius in debug mode

```
# radiusd -X
[root@centos7radius ~]# radtest adtom@eden.com your_password localhost 0 testing123
Sent Access-Request Id 144 from 0.0.0.0:35469 to 127.0.0.1:1812 length 84
        User-Name = "adtom@eden.com"
        User-Password = "your_password"
        NAS-IP-Address = 10.0.0.199
        NAS-Port = 0
        Message-Authenticator = 0x00
        Cleartext-Password = "your_password"
```

```
Received Access-Accept Id 144 from 127.0.0.1:1812 to 0.0.0.0:0 length 20
```

Received Access-Accept should be the response

5. Install and Configure Google Authenticator PAM

Install compile requirements

```
[root@centos7radius ~]# yum install pam-devel make gcc-c++ git
[root@centos7radius ~]# yum install automake autoconf libtool
[root@centos7radius ~]# cd ~

[root@centos7radius ~]# git clone https://github.com/google/google-authenticator-libpam.git
Cloning into 'google-authenticator-libpam'...
remote: Enumerating objects: 796, done.
remote: Total 796 (delta 0), reused 0 (delta 0), pack-reused 796
Receiving objects: 100% (796/796), 538.35 KiB | 381.00 KiB/s, done.
Resolving deltas: 100% (508/508), done.

[root@centos7radius ~]# cd ~/google-authenticator-libpam/

[root@centos7radius google-authenticator-libpam]# ./bootstrap.sh

[root@centos7radius google-authenticator-libpam]# ./configure

[root@centos7radius google-authenticator-libpam]# make

[root@centos7radius google-authenticator-libpam]# make install
```

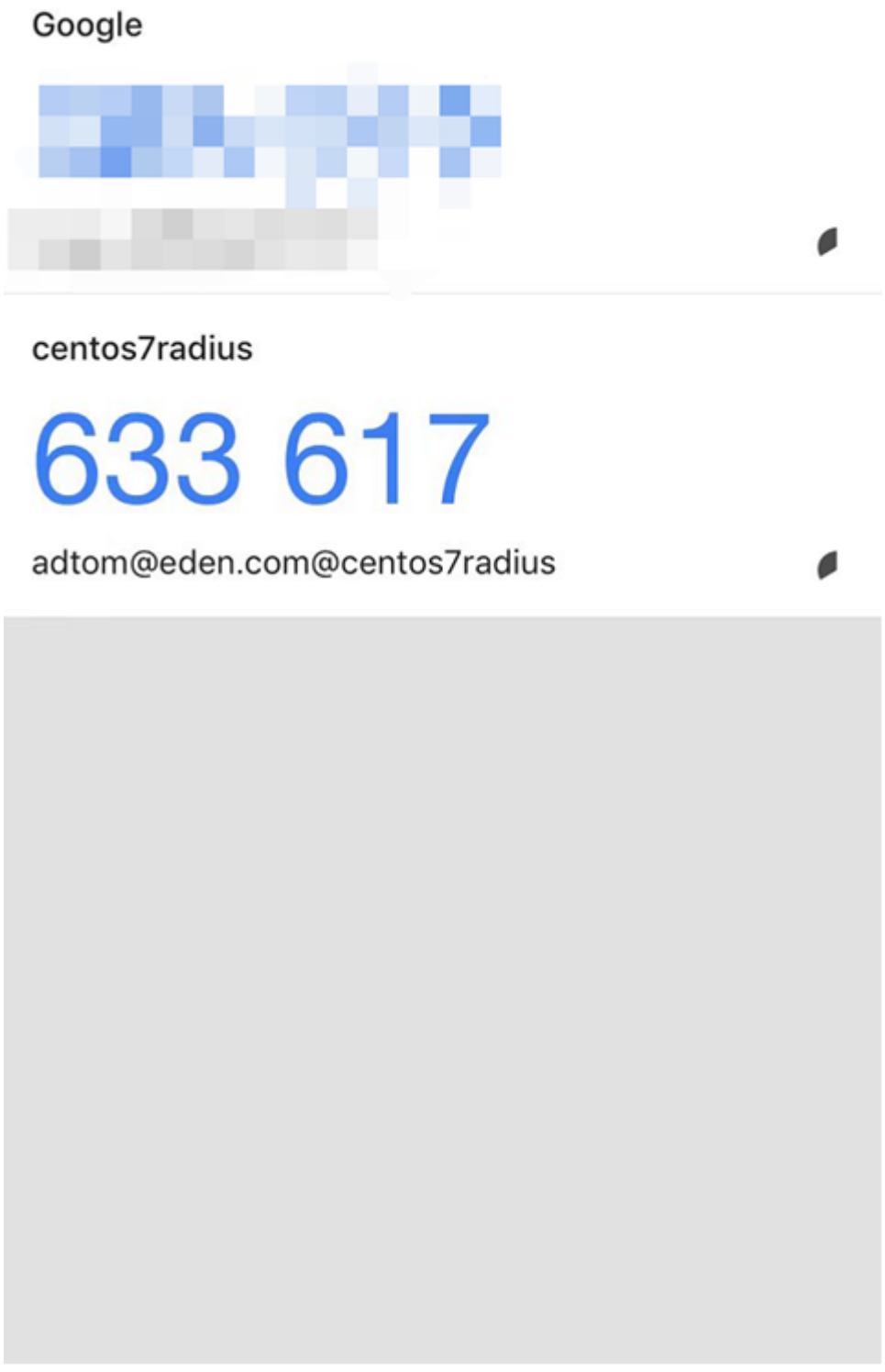
Setup user with google-authenticator

```
[root@centos7radius google-authenticator-libpam]# cd ~
```

```
[root@centos7radius ~]# su - adtom@eden.com
Creating home directory for adtom@eden.com.

[adtom@eden.com@centos7radius ~]$ google-authenticator
Do you want authentication tokens to be time-based (y/n) y
Warning: pasting the following URL into your browser exposes the OTP secret to
Google:
https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://tot
p/adtom@eden.com@centos7radius%3Fsecret%3DWASLQBOJ7SC5CWN3CB
RT62AMOY%26issuer%3Dcentos7radius
```

Open Google Authenticator App on mobile phone and scan the QR Code and input the code shown, in this case, the code is 633617

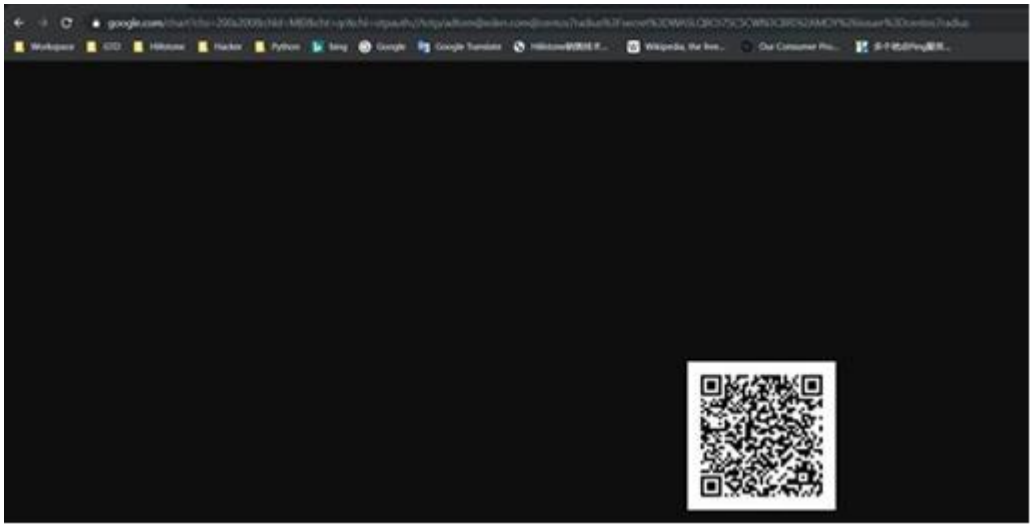


Note:

At this free solution, there is no user self-service portal, the administrator need to generate the QR code on FreeRADIUS server manually for each user and then send the QR code/link to end users via mail for registration at first time.

https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://totp/adtom@eden.com@centos7radius%3Fsecret%3DWASLQBOJ7SC5CWN3CBRT62AMOY%26issuer%3Dcentos7radius

End users can open the link in browser if they have internet access, since this QR code is also stored on Google's server. Such as below:



(during the setup, the administrator can input -1 to skip the code verification < Enter code from app (-1 to skip): -1>)

```
Your new secret key is: WASLQBOJ7SC5CWN3CBRT62AMOY
Enter code from app (-1 to skip): 633617
```

```
Code confirmed
Your emergency scratch codes are:
  43322639
  34705877
  32173950
  41646850
  82907757

Do you want me to update your "/home/adtom@eden.com/.google_authenticator"
file? (y/n) y

Do you want to disallow multiple uses of the same authentication
token? This restricts you to one login about every 30s, but it increases
your chances to notice or even prevent man-in-the-middle attacks (y/n) y

By default, a new token is generated every 30 seconds by the mobile app.
In order to compensate for possible time-skew between the client and the server,
we allow an extra token before and after the current time. This allows for a
time skew of up to 30 seconds between authentication server and client. If you
experience problems with poor time synchronization, you can increase the
window
from its default size of 3 permitted codes (one previous code, the current
code, the next code) to 17 permitted codes (the 8 previous codes, the current
code, and the 8 next codes). This will permit for a time skew of up to 4 minutes
between client and server.
Do you want to do so? (y/n) y

If the computer that you are logging into isn't hardened against brute-force
login attempts, you can enable rate-limiting for the authentication module.
By default, this limits attackers to no more than 3 login attempts every 30s.
Do you want to enable rate-limiting? (y/n) y
```

Responding with y to queries results with

6. Configure PAM

```
[adtom@eden.com@centos7radius ~]$ su root
Password:
[root@centos7radius adtom@eden.com]# vi /etc/pam.d/radiusd
#%PAM-1.0
#auth            include        password-auth
#account        required        pam_nologin.so
#account        include        password-auth
#password     include        password-auth
#session        include        password-auth

auth            requisite       /usr/local/lib/security/pam_google_authenticator.so forward_pass
auth            required        pam_sss.so use_first_pass
account        required        pam_nologin.so
account        include        password-auth
session        include        password-auth
```

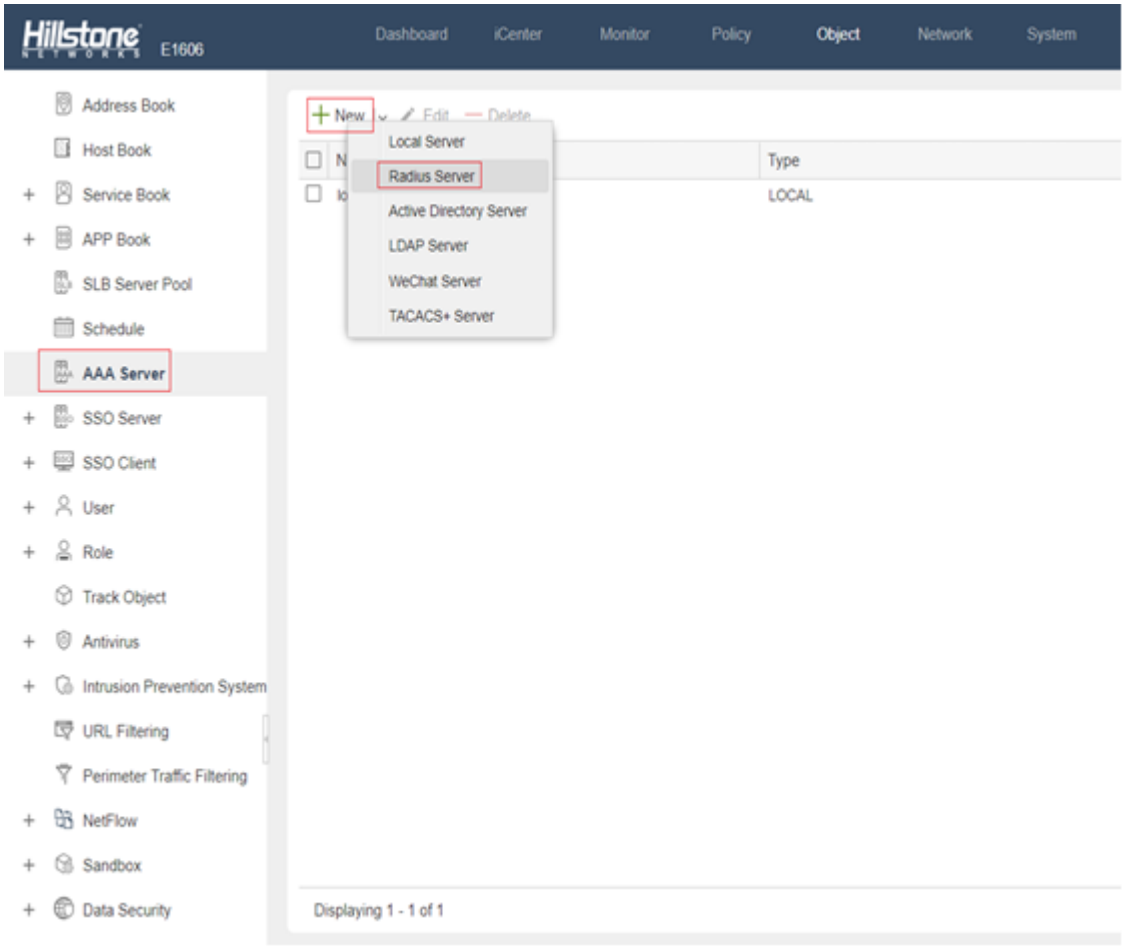
7. Test FreeRADIUS with SSSD & Google Authenticator

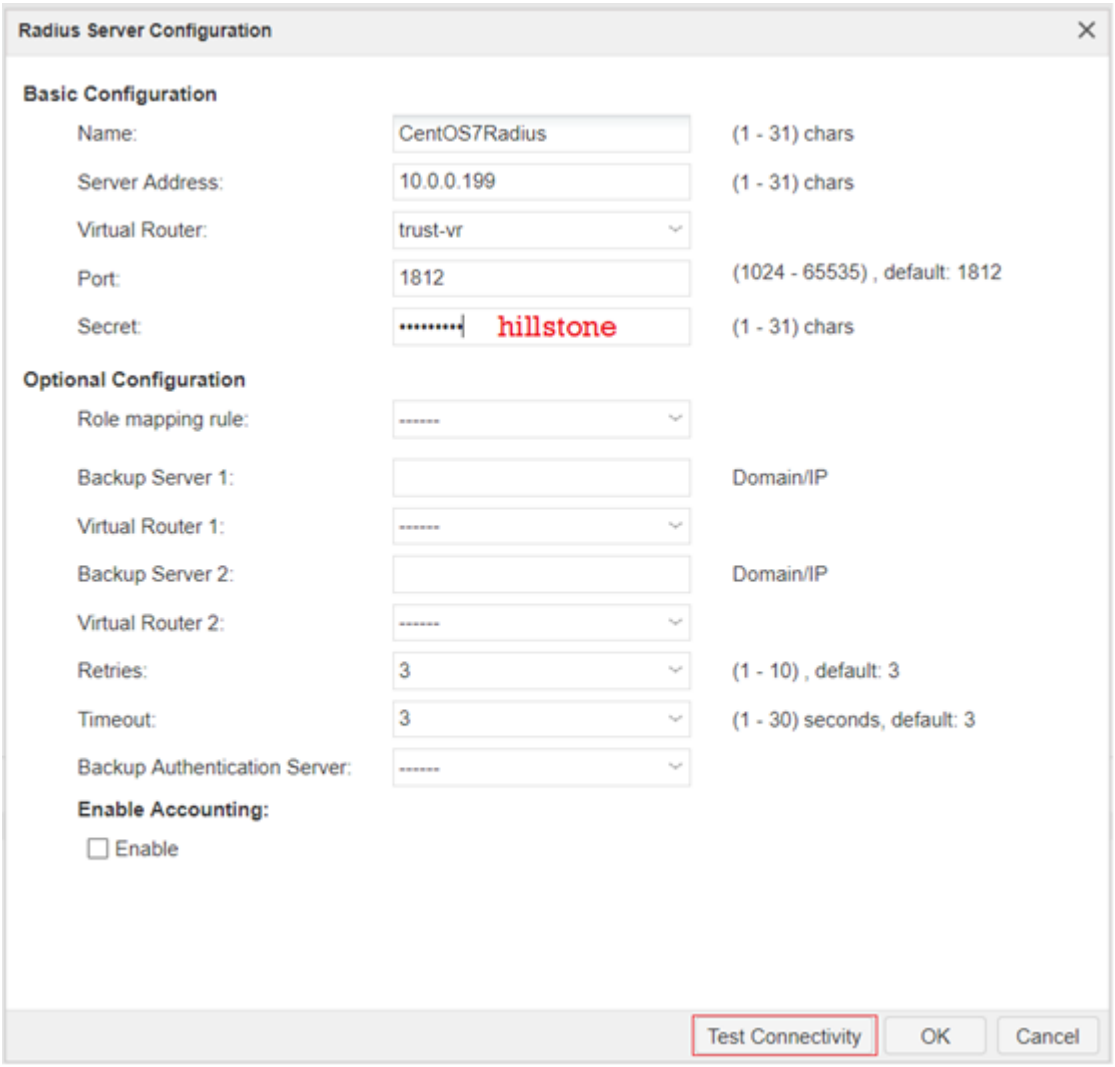
radtest <username> (<active directory pasword><google-authenticator code>) localhost 0 testing123

```
[root@centos7radius adtom@eden.com]# radtest adtom@eden.com your_password077719 localhost 0 testing123
Sent Access-Request Id 121 from 0.0.0.0:60925 to 127.0.0.1:1812 length 100
            User-Name = "adtom@eden.com"
            User-Password = "your_password077719"
            NAS-IP-Address = 10.0.0.199
            NAS-Port = 0
            Message-Authenticator = 0x00
```

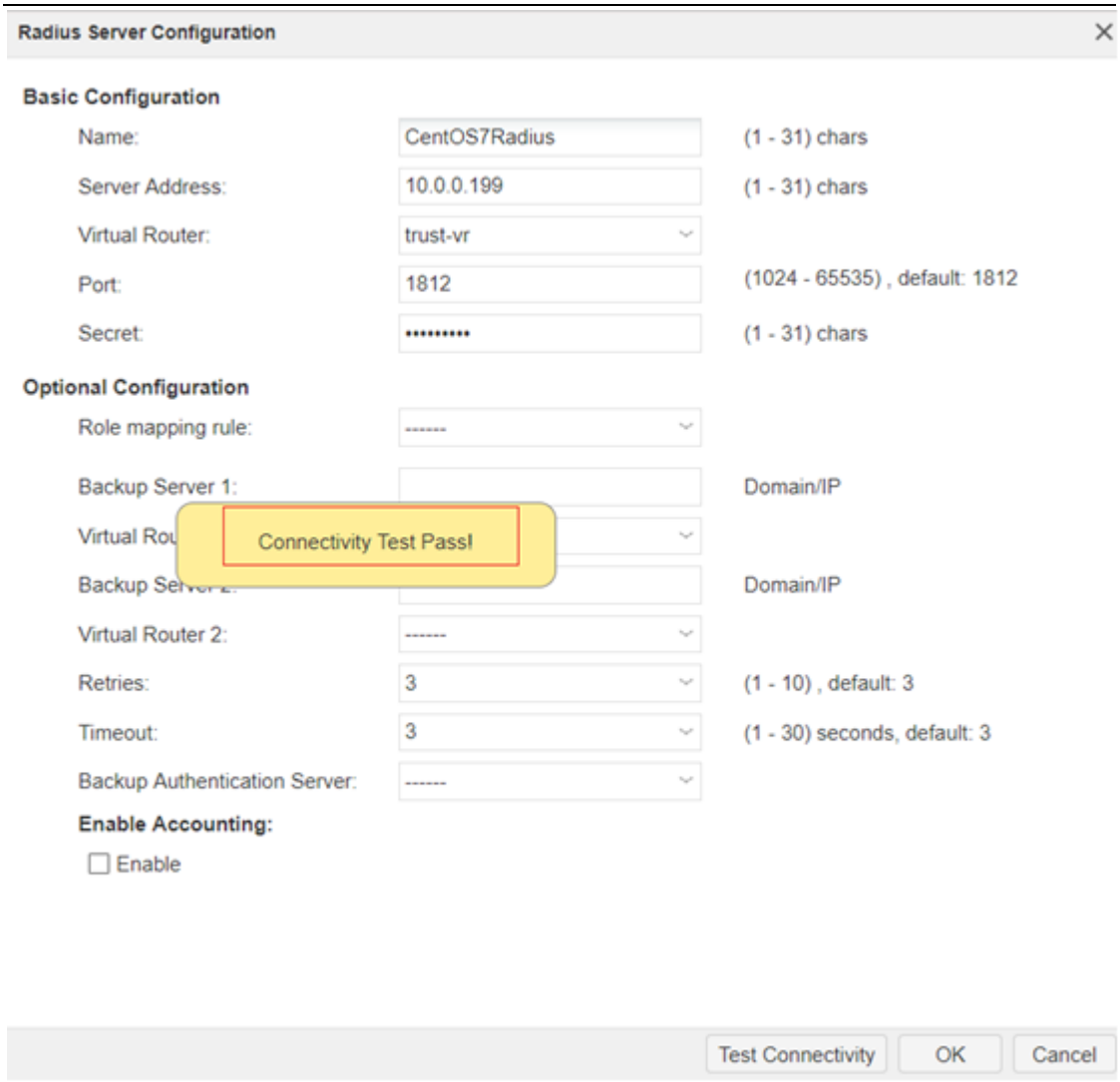
```
        Cleartext-Password = "your_password077719"
Received Access-Accept Id 121 from 127.0.0.1:1812 to 0.0.0.0:0 length 20
```

8. Firewall: Add Radius AAA server in firewall and test authentication

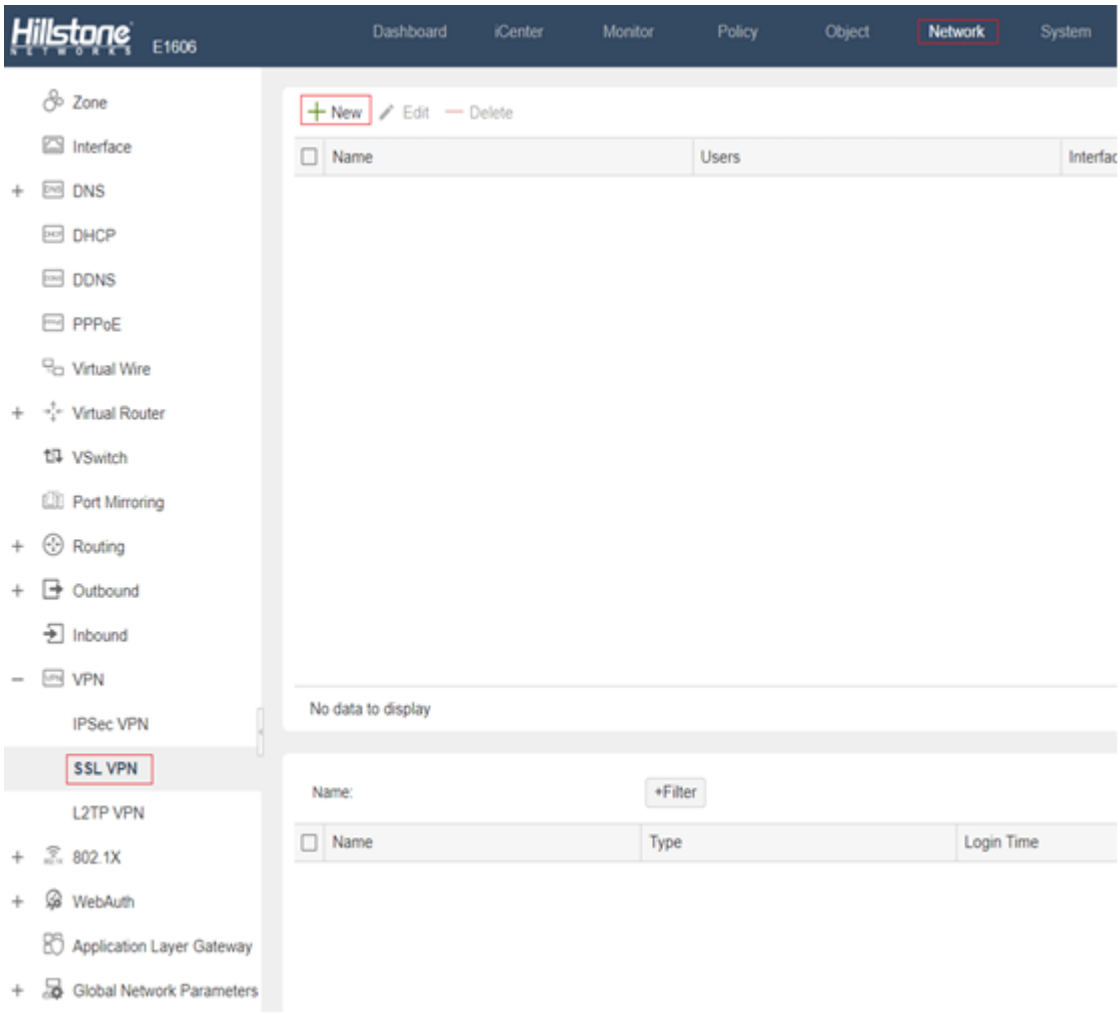


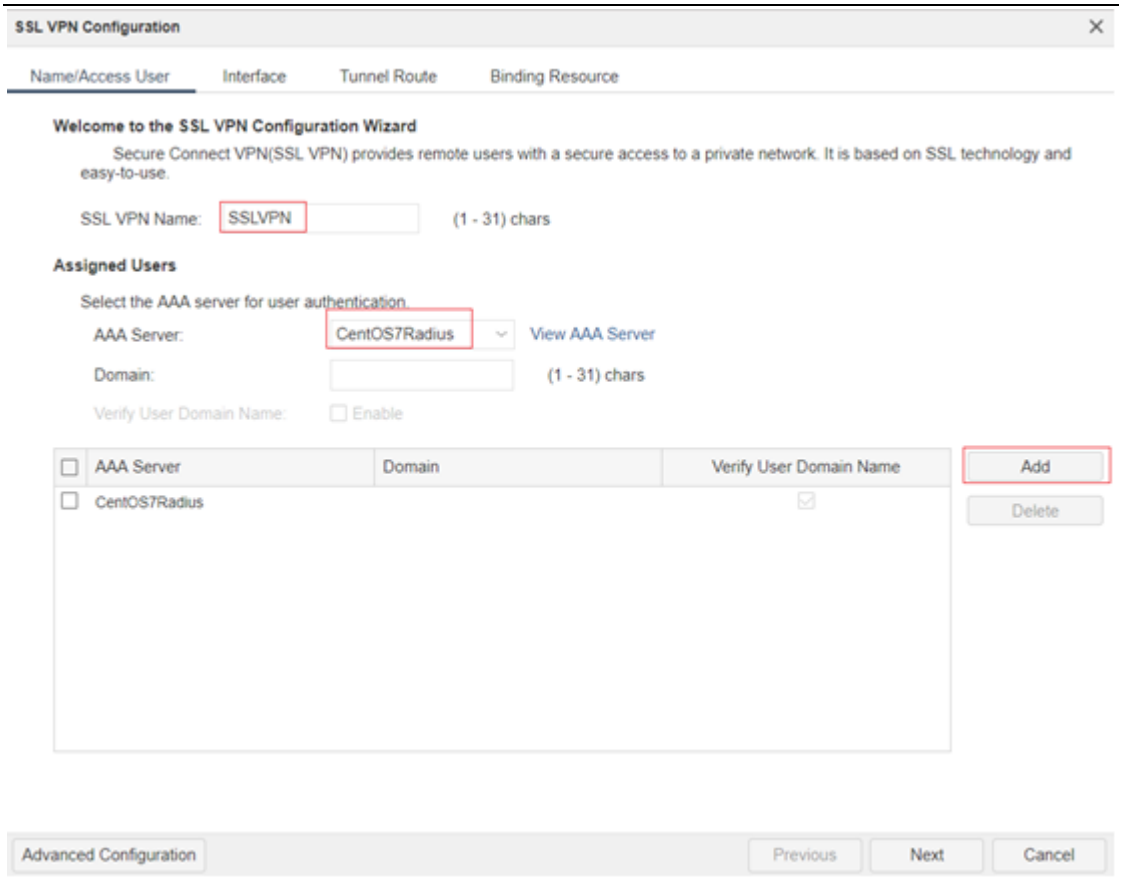


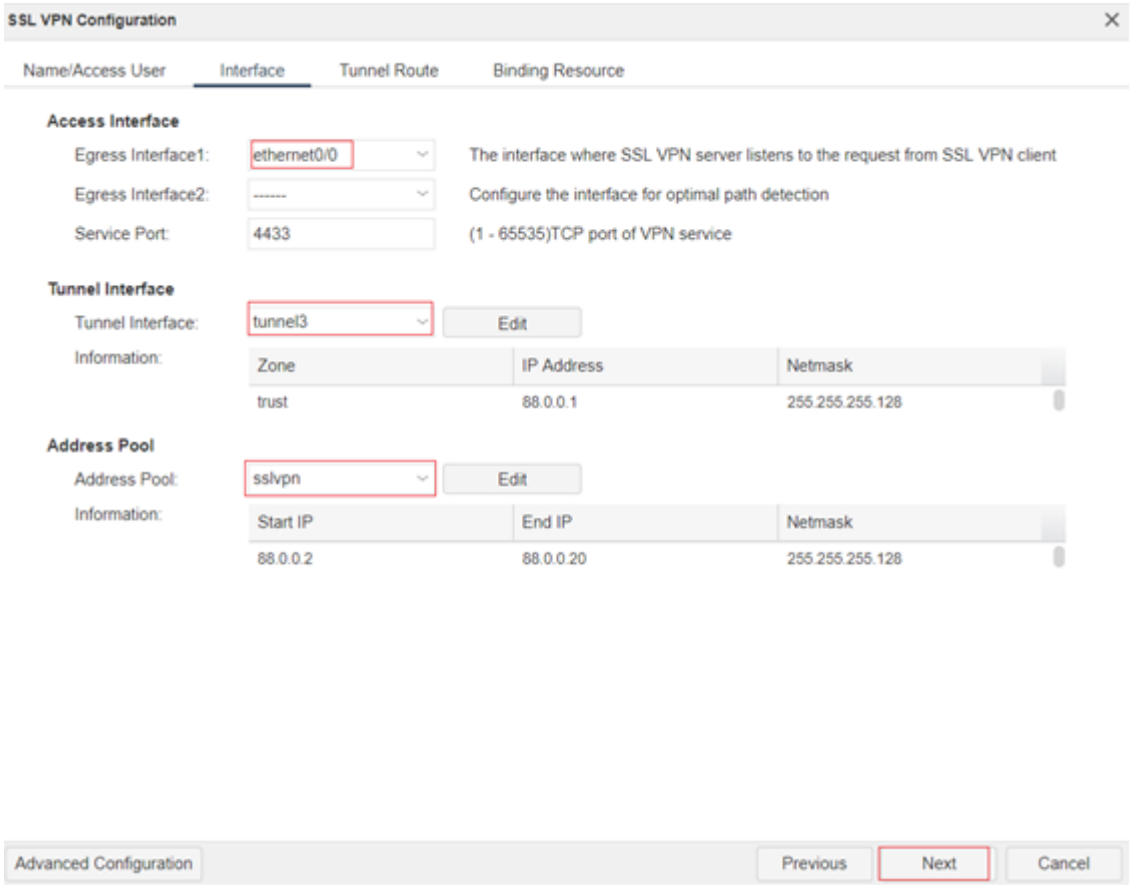
Password is AD account password with code from Google Authenticator App



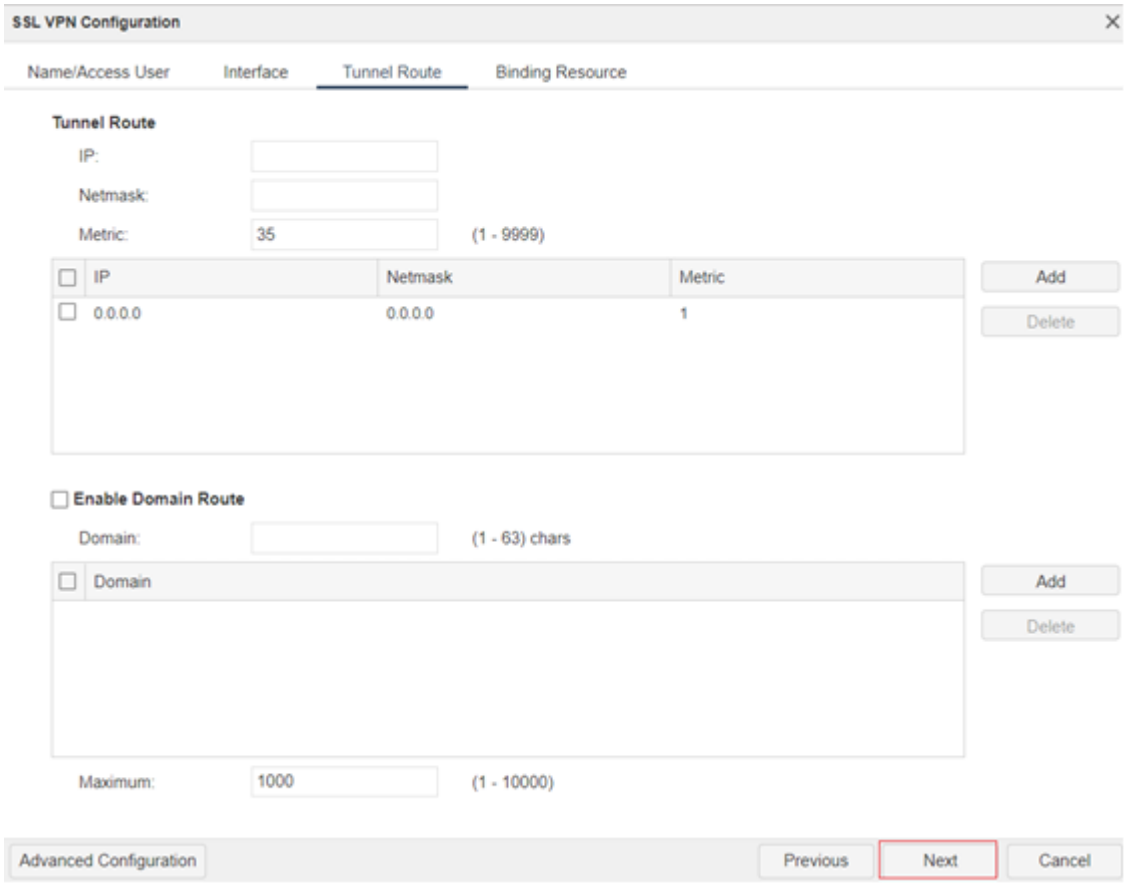
9. Firewall: Configure SSLVPN and use CentOS7Radius as authentication server

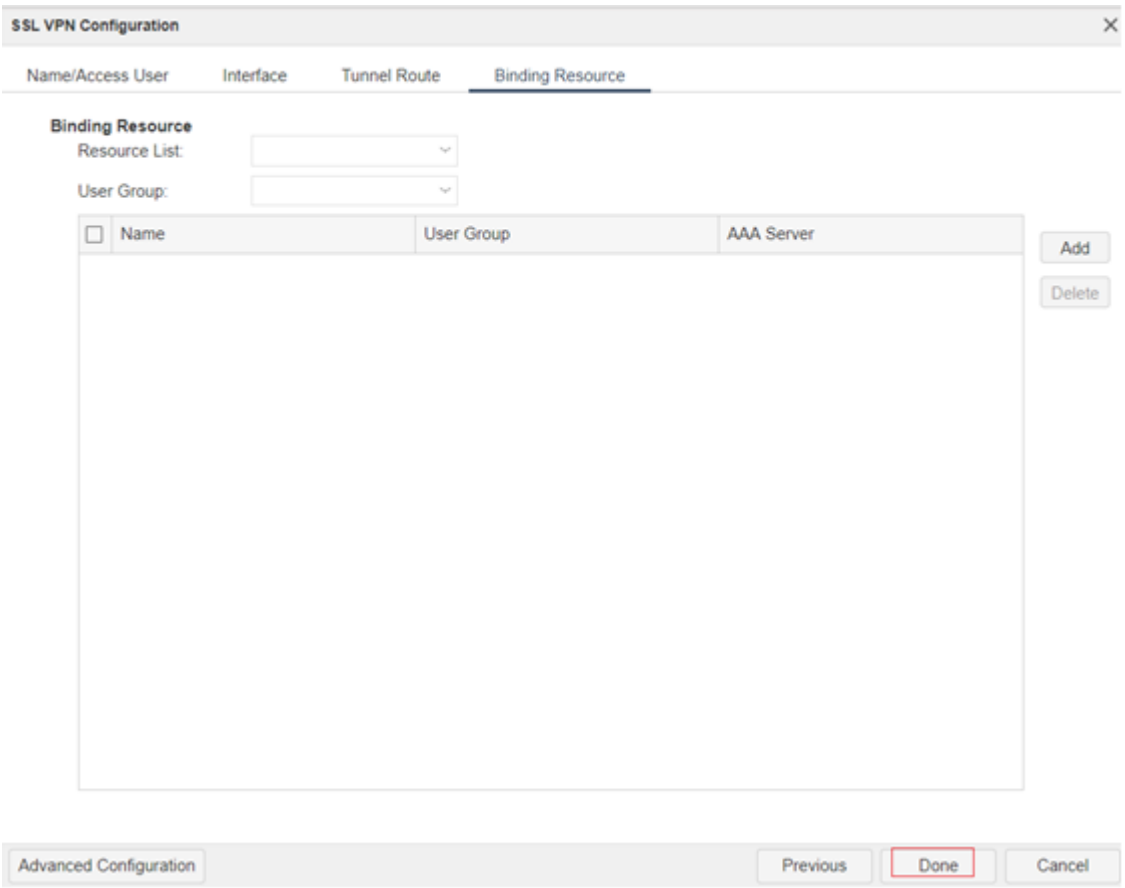

E1606
Dashboard
iCenter
Monitor
Policy
Object
Network
System
Zone
Interface
DNS
DHCP
DDNS
PPPoE
Virtual Wire
Virtual Router
VSwitch
Port Mirroring
Routing
Outbound
Inbound
VPN
IPSec VPN
SSL VPN
L2TP VPN
802.1X
WebAuth
Application Layer Gateway
Global Network Parameters
New
Edit
Delete
Name
Users
No data to display
Name:
+Filter
Name
Type
Login Time


SSL VPN Configuration
Name/Access User
Interface
Tunnel Route
Binding Resource
Welcome to the SSL VPN Configuration Wizard
Secure Connect VPN(SSL VPN) provides remote users with a secure access to a private network. It is based on SSL technology and easy-to-use.
SSL VPN Name:
SSLVPN
(1 - 31) chars
Assigned Users
Select the AAA server for user authentication.
AAA Server:
CentOS7Radius
View AAA Server
Domain:
(1 - 31) chars
Verify User Domain Name:
Enable
AAA Server
Domain
Verify User Domain Name
CentOS7Radius
Add
Delete
Advanced Configuration
Previous
Next
Cancel


SSL VPN Configuration
Name/Access User
Interface
Tunnel Route
Binding Resource
Access Interface
Egress Interface1:
ethernet0/0
The interface where SSL VPN server listens to the request from SSL VPN client
Egress Interface2:

Configure the interface for optimal path detection
Service Port:
4433
(1 - 65535)TCP port of VPN service
Tunnel Interface
Tunnel Interface:
tunnel3
Edit
Information:
Zone
IP Address
Netmask
trust
88.0.0.1
255.255.255.128
Address Pool
Address Pool:
sslvpn
Edit
Information:
Start IP
End IP
Netmask
88.0.0.2
88.0.0.20
255.255.255.128
Advanced Configuration
Previous
Next
Cancel


SSL VPN Configuration
Name/Access User
Interface
Tunnel Route
Binding Resource
Tunnel Route
IP:
Netmask:
Metric:
35
(1 - 9999)
IP
Netmask
Metric
0.0.0.0
0.0.0.0
1
Add
Delete
Enable Domain Route
Domain:
(1 - 63) chars
Domain
Add
Delete
Maximum:
1000
(1 - 10000)
Advanced Configuration
Previous
Next
Cancel



10. Test login on SCVPN Client

When connecting to SSLVPN server, the Password here is AD account password with code from Google Authenticator App.

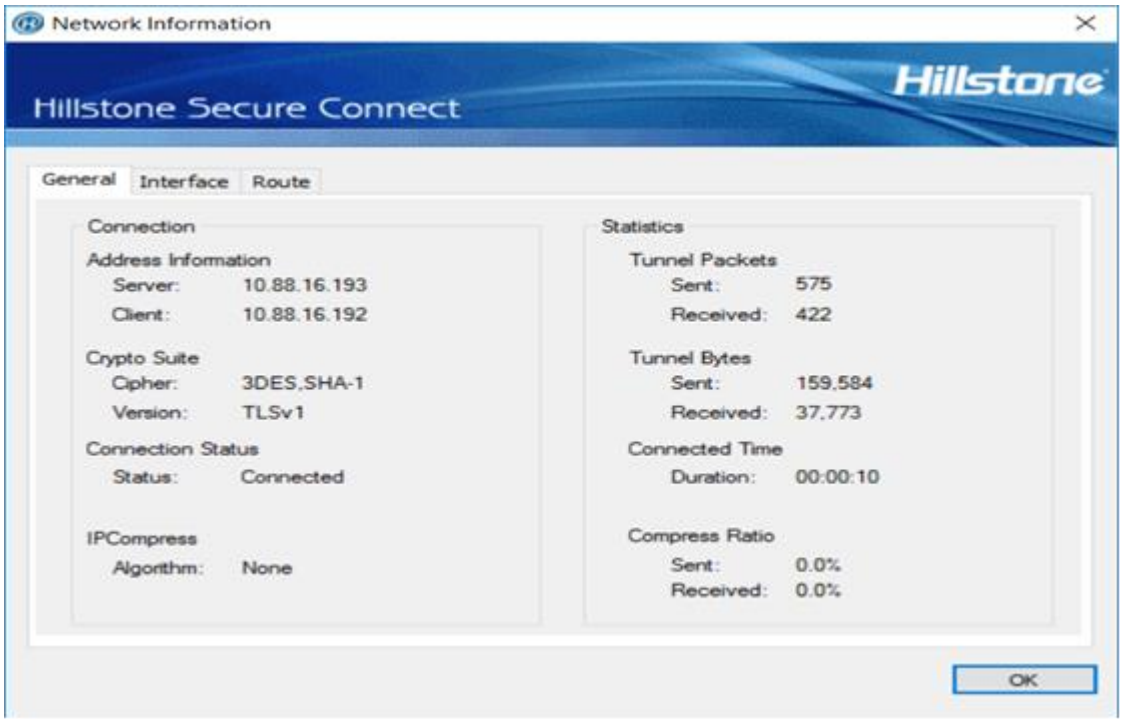
For example, if AD account is adtom@eden.com, AD account password is hillstone@123 and the code in Google Authenticator app is 666 666. The Password you need to input here will be **hillstone@123666666**

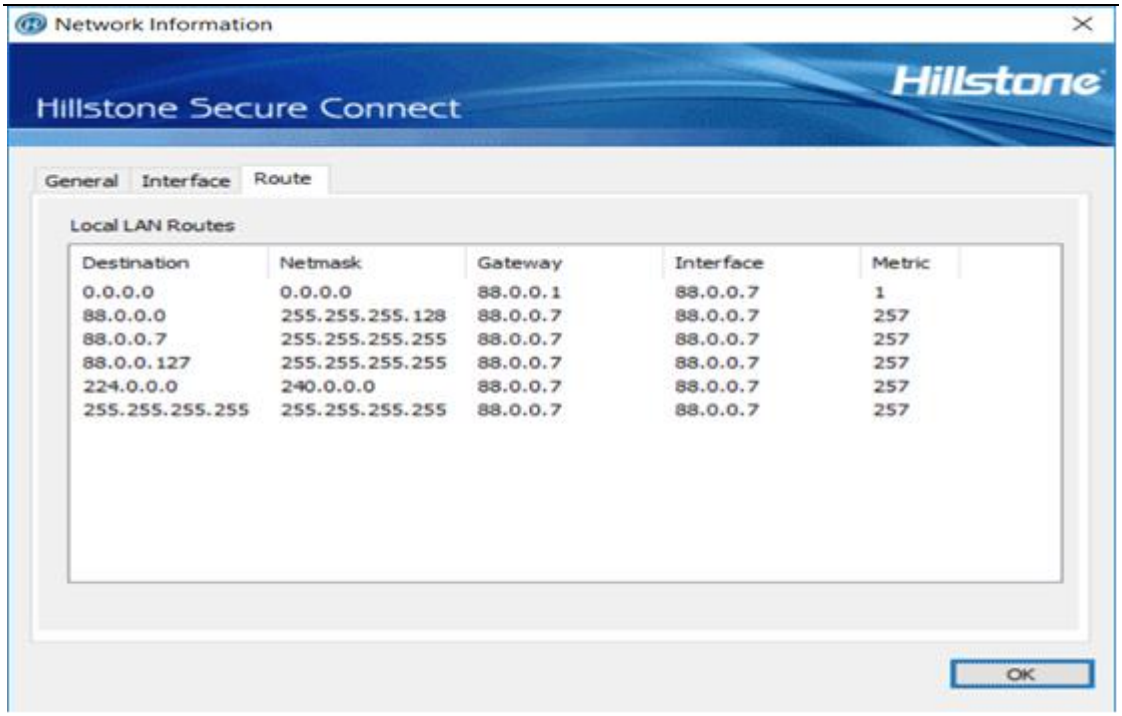
In this solution as we use SSSD to integrate with Win AD, the account information is only stored on AD server, it won't be synchronized to RADIUS server or Firewall.

The authentication process will be:

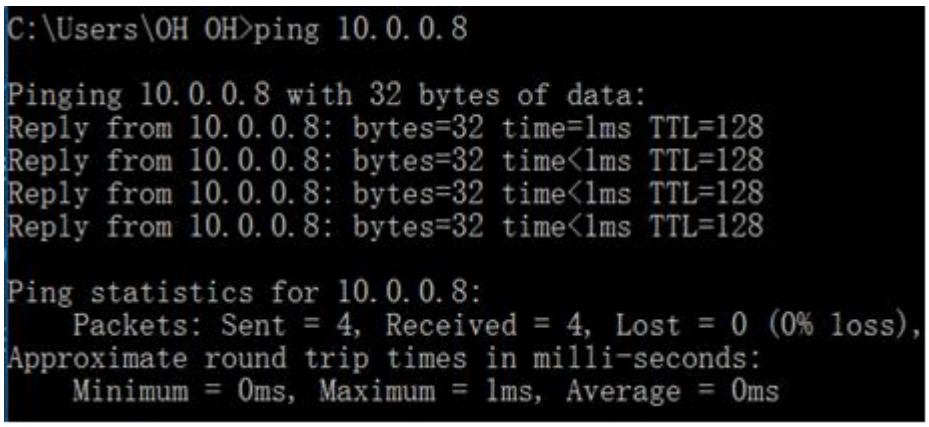
- The firewall forwards the username and password+code to Radius server
- Radius verify the code (2FA)
- Radius verify the password with Win AD server via Kerberos. This process should be similar as a normal login of client in AD domain.
- Radius reply the authentication result to firewall, if passed, the VPN connection is established

Check SSLVPN Connection





Try to ping server in LAN



Note:

There will be an issue in reconnection if SCVPN disconnected. You need to change the password again based on the code on Google Authenticator app.