ACME README - BNSS6

Overview

Our system is to provide a secure network in which users can have access to the web-server and exchange files. With this README file, you can easily set up and use our system.

Prerequisites

It is recommended to deploy each service on separate server in order to build a high robust system. This means you need a set of VMs with CentOS 7 (there could be some slight difference if you use other OS) properly installed.

You need a sub network block and make sure all the servers are connected internally.

Our sub network block in this guide is 192.168.0.0/24.

Setup

PKIs

Public Key Infrastructures is vital to our authentication system. We choose to use `easy-rsa 3`. We will start by building our own root Certificate Authority. For security considerations, you need to place the CA server on a totally isolated network.

Build a Certificate Authority

Download easy-rsa

You will need to enable the Extra Packages for Enterprise Linux (EPEL) repository.

```
sudo dnf install epel-release
```

Then download and install `easy-rsa` package

```
sudo dnf install easy-rsa
```

Prepare a PKI Directory

PKI Directory can help you manage all the certificate and sign requests conveniently.

Execute the following commands

```
mkdir ~/easy-rsa
# soft link the executable script
ln -s /usr/share/easy-rsa/3/* ~/easy-rsa/
# restrict your pki dirctory for security concern.
chmod 700 ~/easy-rsa
# initialize the pki directory
cd ~/easy-rsa
./easyrsa init-pki
```

Now you have a place to store all the PKI information.

Build CA

Before build your own CA, you need to provide some necessary information for your CA.

```
# edit ~/easy-rsa/tars
set_var EASYRSA_REQ_COUNTRY     "Country"
set_var EASYRSA_REQ_PROVINCE    "Province"
set_var EASYRSA_REQ_CITY        "City"
set_var EASYRSA_REQ_ORG         "Organization"
set_var EASYRSA_REQ_EMAIL       "admin@example.com"
set_var EASYRSA_REQ_OU          "Community"
set_var EASYRSA_ALGO            "ec"
set_var EASYRSA_DIGEST          "sha512"
```

The run `./easy-rsa build-ca`, you will be prompted to enter a **Common Name**. This is a very important value, make sure all the Common Names are distinct.

You now have two important files — `~/easy-rsa/pki/ca.crt` and `~/easy-rsa/pki/private/ca.key`, and your CA is ready to be used to sign certificate requests, and to revoke certificates.

Request and Sign Certificates

Request and Sign are general operations to generate a certificate, and are always happened on two hosts to guarantee high security. This can be divided into the follows miner steps:

1. The requesting host provide necessary information to generate a Certificate Sign Request. The CSR include the host's public key and provided information for CA to verify.

   ```
   ./easyrsa gen-req COMMONNAME nopass # you need to provide necessary
   information
   ```

2. The requesting host send CSR to CA in a security way (though CSR contains no secret data like private key), usually by `scp`

   ```
   scp ~/easy-rsa/pki/reqs/COMMONNAME.req user@your_ca_server_ip:/tmp
   ```

3. The CA imports the received CSR, verifies the information and decides whether sign this CSR or not.

   ```
   # the last option can be server/client, depends on the host's identity
   ./easyrsa import-req /tmp/COMMONNAME.req server
   ./easyrsa sign-req server COMMONNAME
   ```

4. If CA signs the CSR, the CA's certificate and the signed requesting host's certificate are sent back to the requesting host. And now the host can make use of the certificate.

   ```
   scp pki/issued/server.crt user@your_vpn_server_ip:/tmp
   scp pki/ca.crt user@your_vpn_server_ip:/tmp
   ```

Revoke Certificates

We need revoke one host's certificate if it's compromised to decrease the security impact. In our realizations, the Certificate Revocation List is required to distribute to different service manually.

On the Certificate Authority Server:

1. Revoke the certificate with the `./easyrsa revoke client_name` command.
2. Generate a new CRL with the `./easyrsa gen-crl` command.
3. Transfer the updated `crl.pem` file to the server or servers that rely on your CA, and on those systems copy it to the required directory or directories for programs that refer to it.
4. Restart any services that use your CA and the CRL file.

File Server

We use Seafile Pro as our File Server. Follow the instruction and download Seafile Pro from here. Since this is an opensource software, the official guideline is sufficient to finish installation. What we focus here is the server customization and configuration.

File Server Configuration

After installation, you can change to the `seafile-server-latest` directory and execute `./seafile.sh start` and `./seahub.sh start` to start the file service. By default, `seafile` is at port 8000 and `seahub` is at port 8082. You can use `your-ip-address:8082` to visit the web interface of Seafile (Seahub).

Secure the File Transmission

The next step is to use an Apache reverse proxy server at the same VM to enable us to visit Seafile by `https://your-ip-address`. By this way we can encrypt our file data traffic using SSL/TLS protocol and keep your data secret from others. What's more, we only need open port 443 to others instead of 8000 and 8082 after doing this.

> Though we will deploy an Apache Server which dedicates to encapsulate all the traffic with SSL later, our experiences tell us we need an extra https proxy here for Seafile. Otherwise there may be some mixed-contents afterwards. This depends on the specific implementation of File Server Software. For Seafile, it's necessary to do this operation.

To setup the local HTTPS Apache reverse proxy, we first need to request a key/certificate pair from the Certificate Authority. Please refer to Request and Sign Certificates to finish this step.

Then you need to have your apache server installed and make sure the following apache mods are enabled.

```
mod_proxy
mod_proxy_http
mod_proxy_balancer
mod_lbmethod_byrequests
mod_ssl
```

The first four mods should be naturally enabled with Apache server. You may need to install `mod_ssl` manually. Use this command `sudo yum install mod_ssl`. This mod is automatically enabled once successfully installed. Check with `httpd -M`.

Next edit the virtual host config file at `/etc/httpd/conf.d/ssl.conf`

```
# /etc/httpd/conf.d/ssl.conf
<VirtualHost *:443>
  ServerName your-domain-name
  DocumentRoot /var/www/html

  SSLEngine On
  SSLCertificateFile /path/to/cacert.pem
  SSLCertificateKeyFile /path/to/privkey.pem

  Alias /media  /home/user/haiwen/seafile-server-latest/seahub/media

  <Location /media>
    Require all granted
  </Location>

  RewriteEngine On

  # seafile fileserver
  ProxyPass /seafhttp http://127.0.0.1:8082
  ProxyPassReverse /seafhttp http://127.0.0.1:8082
  RewriteRule ^/seafhttp - [QSA,L]

  # seahub
  SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
  ProxyPreserveHost On
  ProxyPass / http://127.0.0.1:8000/
  ProxyPassReverse / http://127.0.0.1:8000/
</VirtualHost>
```

Up to now your Apache server is ready to use. Before restart your Seafile server, do the following changes to the Seafile config file.

1. Change the `SERVICE_URL` in `ccnet.conf` to

   ```
   SERVICE_URL = https://your-domain-name
   ```

2. Change the `FILE_SERVER_ROOT` in `seahub_settings.py` to

   ```
   FILE_SERVER_ROOT = 'https://your-domain-name/seafhttp'
   ```

Now restart the Seafile, Seahub and Apache server and you shall visit the file server with `https://your-domain-name`.

Web Server

We won't focus on the detailed contents on the web server. All we need in this step is an Apache Server that can provide a web service. We can do this by merely two commands.

```
# Install Apache
sudo yum install httpd
# Start Apache
sudo service httpd start
```

Try to visit `http://your-ip-address` and you will get a default test page.

OpenVPN Server

VPN Server plays an important role in security. We must make sure only authenticated user can establish a VPN connection.

Our Solution

We choose OpenVPN as our VPN implementation and deploy two VPN server instance on one VM in order to provide two authentication method.

In this part we focus on our OpenVPN Server/Client configuration. If you haven't install OpenVPN yet, please refer to General Installation.

Using Certificate

The first VPN server instance on the VM listens to port 1194/udp and expects Certificate Authentication. This instance doesn't need extra configuration after you install the OpenVPN itself. The `server.conf` and `client1.ovpn` are shown below.

```
# server.conf
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key  # This file should be kept secret
dh none
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 208.67.222.222"
push "dhcp-option DNS 208.67.220.220"
keepalive 10 120
tls-crypt ta.key
cipher AES-256-GCM
auth SHA256
user nobody
group nobody
persist-key
persist-tun
```

```
status openvpn-status.log
verb 3
explicit-exit-notify 1
```

```
# client1.ovpn
client
dev tun
proto udp
remote 3.129.108.5 1194
resolv-retry infinite
nobind
user nobody
group nobody
persist-key
persist-tun
remote-cert-tls server
cipher AES-256-GCM
auth SHA256
verb 3
key-direction 1
<ca>
embedded ca cert goes here
</ca>
<cert>
embedded cert goes here
</cert>
<key>
embedded key goes here
</key>
<tls-crypt>
embedded tls key goes here
</tls-crypt>
```

Using Username/Password with 2FA

The second VPN server instance listens to port 1194/tcp and expects username/password with 2FA authentication.

We need some configuration on the server end in order to enable two factor authentication.

First install the google-authenticator plugin.

```
LC_ALL=C yum -y groupinstall "Development Tools"
yum -y install pam-devel
mkdir /usr/src
cd /usr/src
git clone https://github.com/google/google-authenticator-libpam
cd google-authenticator-libpam
./bootstrap.sh
./configure
make && make install
```

Then create `/etc/pam.d/openvpn` file and input these lines.

```
# vi /etc/pam.d/openvpn
# google auth
auth        required    /usr/local/lib/security/pam_google_authenticator.so
forward_pass
account     required    pam_nologin.so
account     include     system-auth use_first_pass
password    include     system-auth
session     include     system-auth
```

Now we binds the user with google authenticator. Just type `google-authenticator` and follow the instruction.

The `server-2.conf` and `client2.ovpn` are shown below.

```
# server-2.conf
port 1194
proto tcp
dev tun
ca ca.crt
cert server.crt
key server.key  # This file should be kept secret
dh none
server 10.16.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 208.67.222.222"
push "dhcp-option DNS 208.67.220.220"
keepalive 10 120
tls-crypt ta.key
cipher AES-256-GCM
auth SHA256
user nobody
group nobody
persist-key
persist-tun
status openvpn-status.log
verb 3
#-----------------for passwd&2fa-----------#
script-security 3
plugin /usr/share/openvpn/plugin/lib/openvpn-auth-pam.so /etc/pam.d/openvpn
verify-client-cert none
username-as-common-name
```

```
# client2.ovpn
client
dev tun
proto tcp
remote 3.129.108.5 1194
resolv-retry infinite
nobind
user nobody
group nobody
persist-key
persist-tun
remote-cert-tls server
cipher AES-256-GCM
```

```
auth SHA256
verb 3
key-direction 1
<tls-crypt>
embedded tls key goes here
</tls-crypt>
auth-user-pass # for passwd&2fa
```

General Installation

OpenVPN have a regulation about where to place the key and how to name the file, so we will include all the details from the key generation.

Installing OpenVPN and Easy-RSA

First you need to install easy-rsa on your server:

```
sudo dnf install epel-release
sudo dnf install openvpn easy-rsa
```

Then you should make your own directory and then create a sym-link:

```
mkdir ~/easy-rsa
ln -s /usr/share/easy-rsa/3/* ~/easy-rsa/
```

And then you should change the owner and restrict access to this directory:

```
sudo chown centos ~/easy-rsa
chmod 700 ~/easy-rsa
```

Creating a PKI for OpenVPN

First you need to create a new file:

```
cd ~/easy-rsa
nano vars
```

and add two lines in this file:

```
set_var EASYRSA_ALGO "ec"
set_var EASYRSA_DIGEST "sha512"
```

which means you will use ECC Elliptic Curve Cryptography (ECC) to generate keys and secure signatures for your clients and OpenVPN server

Finally use this command to create the PKI directory:

```
./easyrsa init-pki
```

Creating Certificate Request and Private Key for VPN server

```
./easyrsa gen-req server nopass
sudo cp /home/centos/easy-rsa/pki/private/server.key /etc/openvpn/server/
```

After that you will get a `server.req` file and a `server.key` file.

Signing the OpenVPN Server’s Certificate Request

First you need to use `scp` to copy the `server.req` to the CA server, and then use your CA server to sign the request and generate server.crt:

```
cd ~/easy-rsa
./easyrsa import-req /tmp/server.req server
./easyrsa sign-req server server
```

Here the request type is `server` and the name is also `server`.

Then you can copy `server.crt` and `ca.crt` to your VPN server:

```
sudo cp /tmp/{server.crt,ca.crt} /etc/openvpn/server
```

Configure the control channel packets encryption

```
cd ~/easy-rsa
openvpn --genkey --secret ta.key
sudo cp ta.key /etc/openvpn/server
```

Now you have the pre-shared key for control channel packets encryption

Generating a client certificate and key pair

First you have to create a directory to store the client certificate and key file:

```
mkdir -p ~/client-configs/keys
chmod -R 700 ~/client-configs
```

Then you should generate the request for your client, and sign it with your CA server. It is very similar with previous case (sign server request), but here you should use client type:

```
cd ~/easy-rsa
./easyrsa gen-req client1 nopass
cp pki/private/client1.key ~/client-configs/keys/
```

Copy this request to your CA and change to your CA server, then do these commands to sign the request:

```
cd ~/easy-rsa
./easyrsa import-req /tmp/client1.req client1
./easyrsa sign-req client client1
```

Copy the `ca.crt` and `ta.key` files to the `~/client-configs/keys/` directory:

```
cp ~/easy-rsa/ta.key ~/client-configs/keys/
sudo cp /etc/openvpn/server/ca.crt ~/client-configs/keys/
sudo chown centos.centos ~/client-configs/keys/*
```

Now your server and client's certificates and keys have all been generated and are stored in the appropriate directories on your OpenVPN server.

Configuring your OpenVPN server

First we need to copy the `server.conf` as our example:

```
sudo cp /usr/share/doc/openvpn/sample/sample-config-files/server.conf
/etc/openvpn/server/
sudo nano /etc/openvpn/server/server.conf
```

Then we need to change several lines of this file like that:

```
;tls-auth ta.key 0
tls-crypt ta.key

;cipher AES-256-CBC
cipher AES-256-GCM

auth SHA256

;dh dh2048.pem
dh none

user nobody
group nobody
```

Adjusting the network and firewall configuration

```
sudo nano /etc/sysctl.conf
```

Then add the following line at the top of the file:

```
net.ipv4.ip_forward = 1
```

Then add OpenVPN service on the firewall active zone:

```
sudo firewall-cmd --zone=trusted --add-interface=tun0
sudo firewall-cmd --permanent --zone=trusted --add-interface=tun0
sudo firewall-cmd --permanent --add-service openvpn
sudo firewall-cmd --permanent --zone=trusted --add-service openvpn
sudo firewall-cmd --reload
```

Then create the specific masquerade rule for your OpenVPN subnet:

```
DEVICE=$(ip route | awk '/^default via/ {print $5}')
sudo firewall-cmd --permanent --direct --passthrough ipv4 -t nat -A POSTROUTING
-s 10.8.0.0/24 -o $DEVICE -j MASQUERADE
sudo firewall-cmd --reload
```

Now we can start our service,and it should start up at boot:

```
sudo systemctl -f enable openvpn-server@server.service
sudo systemctl start openvpn-server@server.service
```

Making the Client Configuration

This step is very similar with the configuration of server. First we need to copy `client.conf` as an example:

```
mkdir -p ~/client-configs/files
cp /usr/share/doc/openvpn-2.4.10/sample/sample-config-files/client.conf
~/client-configs/base.conf
nano ~/client-configs/base.conf
```

Then we need to change several lines of this file like that:

```
remote 3.129.108.5 1194

proto udp

user nobody
group nobody

;ca ca.crt
;cert client.crt
;key client.key

;tls-auth ta.key 1

cipher AES-256-GCM
auth SHA256

key-direction 1

; script-security 2
; up /etc/openvpn/update-resolv-conf
; down /etc/openvpn/update-resolv-conf
```

Next, we will create a script which will compile your base configuration with the relevant certificate, key, and encryption files. Type `nano ~/client-configs/make_config.sh` and edit

```
#!/bin/bash

# First argument: Client identifier

KEY_DIR=~/client-configs/keys
OUTPUT_DIR=~/client-configs/files
BASE_CONFIG=~/client-configs/base.conf

cat ${BASE_CONFIG} \
<(echo -e '<ca>') \
${KEY_DIR}/ca.crt \
<(echo -e '</ca>\n<cert>') \
${KEY_DIR}/${1}.crt \
<(echo -e '</cert>\n<key>') \
${KEY_DIR}/${1}.key \
<(echo -e '</key>\n<tls-crypt>') \
${KEY_DIR}/ta.key \
<(echo -e '</tls-crypt>') \
> ${OUTPUT_DIR}/${1}.ovpn
```

```
chmod 700 ~/client-configs/make_config.sh
```

This script will automatically generate client configuration file with all the certificate and key files.

Finally, you can generate a config file for client1 and the output is a file like `client1.ovpn`

```
cd ~/client-configs
./make_config.sh client1
```

Then import it in your client application and you can use it to connect to your VPN server!

Authentication Apache Reverse Proxy with SSL

When employees enter the internal network after connecting to the VPN, we need another one authentication step to prevent potential attacks like insiders intrusion. We here provide two authentication approaches, i) authenticate with client certificates, ii) authenticate with Kerberos. Since we are using the very original Kerberos without aggregating it into a SSO authentication web application (like KTH SSO), we will use two `virtualhosts` on one Apache Server to take different authentication requests.

Encrypting all the traffic on the fly is also important, and we will implement HTTPS on both `virtualhosts`.

Certificate Authentication

In this way, the client needs a certificate to authenticate himself to the web server. To implement this functionality, we create a `virtualhost` and let it listen to port 443.

```
Listen 443
<VirtualHost *:443>
        DocumentRoot /var/www/html
        ServerName web.bnss6.com
        SSLEngine On

        SSLVerifyClient require
        SSLVerifyDepth 10
        SSLCACertificateFile "/home/centos/easy-rsa/pki/ca.crt"
</VirtualHost>
```

This will tell Apache server to listen to port 443 and verify client's certificate using CA's certificate provided in `SSLCACertificateFile`.

Employees can request certificate from CA and generate a `.p12` certificate file. Use this command:

```
openssl pkcs12 -export -inkey selfsigned-cli.key -in selfsigned-cli.crt -out
selfsigned-cli.p12
```

Then they need to import this `.p12` file to their web browser and start browsing!

Kerberos Authentication

In this way, the client needs a username and a password to authenticate himself to the web server. To implement this, we first need a Kerberos Server.

Setup Kerberos Server

Install the package first

```
yum -y install krb5-server krb5-libs krb5-workstation
```

Then you need to configure three config files.

```
# /var/kerberos/krb5kdc/kdc.conf
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
 BNSS6.COM = {
  #master_key_type = aes256-cts
  acl_file = /var/kerberos/krb5kdc/kadm5.acl
  dict_file = /usr/share/dict/words
  admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
  supported_enctypes = aes256-cts:normal aes128-cts:normal des3-hmac-sha1:normal
arcfour-hmac:normal camellia256-cts:normal camellia128-cts:normal des-hmac-
sha1:normal des-cbc-md5:normal des-cbc-crc:normal
 }
```

```
# /var/kerberos/krb5kdc/kadm5.acl
*/admin@BNSS6.COM       *
```

`kadm5.acl` is a file to control system privilege.

1. The first parameter means the user you want to grant privilege to.
2. The second parameter means the privilege you want to grant. ***** means all privilege.

```
# /etc/krb5.conf
# Configuration snippets may be placed in this directory as well
includedir /etc/krb5.conf.d/

[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 dns_lookup_realm = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
 rdns = false
 pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
 default_realm = BNSS6.COM
 #default_ccache_name = KEYRING:persistent:%{uid}

[realms]
BNSS6.COM = {
        kdc = krb.bnss6.com:88
        admin_server = krb.bnss6.com:749
        default_domain = BNSS6.COM
}

[domain_realm]
 .bnss6.com = BNSS6.COM
 bnss6.com = BNSS6.COM

[kdc]
 profile = /var/kerberos/krb5kdc/kdc.conf
```

The `krb5.conf` is the main config file. It includes log, domain realm, the ticket lifetime and so on.

After editing these files, you can execute this command to initialize a KDC database where you store all the passwords in. You will be prompt to set a admin password. Carefully remember this!

```
kdb5_util create -r BNSS6.COM -s
```

Then you can start the service with `systemctl start krb5kdc` and `systemctl start kadmin`.

The next step is to add a admin user, use the command `kadmin.local` and then type `addprinc root/admin`. Follow the instructions.

At this stage your Kerberos Server is successfully set up.

Setup Kerberos Client Environment on Apache Server

You need to setup the Kerberos Environment before implementing Apache Kerberos AUTH.

Likewise, install the package first

```
yum install -y krb5-lib krb5-workstation
```

Then the simplest way to configure the Kerberos client is to copy the `krb5.conf` from the server to this client. Use `scp` !

Now try use `kinit` to get a ticket and `klist` to see granted tickets.

Setup Apache Virtual Host

We will create a new `virtualhost` and let it listen to port 444. By this way, if the client want to use Certificate Authentication, he can simply visit 443 port. Otherwise if he want to use Kerberos Authentication, he can instead visit 444 port.

Before creating the `virtualhost` config file, we need to enable `auth_gssapi_module` mod. And then create a new Kerberos user `HTTP/your-domain-name` and its corresponding `keytab` file.

```
kadmin -p bofh/admin -q "addprinc -randkey HTTP/www.example.com"
kadmin -p bofh/admin -q "ktadd -k /etc/httpd/http.keytab HTTP/www.example.com"
# make sure apache can access this file
chown apache /etc/httpd/http.keytab
```

Then you can create the `virtualhost`.

```
Listen 444
<VirtualHost *:444>
        DocumentRoot /var/www/html
        ServerName web.bnss6.com
        SSLEngine On
        <Location />
                AuthType GSSAPI
                AuthName "BNSS6 | GSSAPI SSO Login"
                GssapiCredStore keytab:/etc/httpd/http.keytab
                Require valid-user
        </Location>
</VirtualHost>
```

Reverse Proxy

Currently we have two service to employees: The web server and the Seafile server. We want the user can access the two service by one URL, so we will need do reverse proxy to 'forward' different requests to different services based on the path.

We want to implement this feature no matter by which method employee is authenticated. Thus we will configure it in `/etc/conf/httpd.conf`. Editing this file will influence all the `virtualhosts`.

In `/etc/conf/httpd.conf`, add the following lines in the end.

```
ProxyPass    /web/ http://192.168.0.91/     # The IP address of the Web server
ProxyPassReverse  /web/ http://192.168.0.91/# for redirection

ProxyPass    / https://192.168.0.43/        # The IP address of the Seafile
server
ProxyPassReverse  / https://192.168.0.43/   # for redirection
```

SSL Encryption

We want to encrypt all our traffic in the Internet. You need request a certificate for this Apache Server first. Refer to Request and Sign Certificates .

Then add the following lines in the `/etc/conf/httpd.conf`.

```
SSLProxyEngine On
SSLProxyCheckPeerName Off
SSLProxyCheckPeerCN Off
ProxyRequests Off
SSLCertificateFile "/home/centos/easy-rsa/pki/tls/ssl-server.crt"
SSLCertificateKeyFile "/home/centos/easy-rsa/pki/tls/ssl-server.key"
```

Radius Server

We will use FreeRadius to implement Radius Server.

Download

```
#download freeradius,mysql,mariadb-core
yum install -y freeradius freeradius-utils freeradius-mysql mariadb-server
#verify correctly install
rpm -qa |grep mariadb
rpm -qa |grep freeradius
```

Initialize

```
#start and enable the service when boot
systemctl start mariadb.service
systemctl enable mariadb.service
systemctl start radiusd.service
systemctl enable radiusd.service
```

Configuration

```
# configure the radiusd.service file
vi /usr/lib/systemd/system/radiusd.service

[Unit]
Description=FreeRADIUS high performance RADIUS server.
After=syslog.target network.target ipa.service dirsrv.target krb5kdc.service
mariadb.service

[Service]
Type=forking
PIDFile=/var/run/radiusd/radiusd.pid
ExecStartPre=-/bin/chown -R radiusd.radiusd /var/run/radiusd
ExecStartPre=/usr/sbin/radiusd -C
ExecStart=/usr/sbin/radiusd -d /etc/raddb
ExecReload=/usr/sbin/radiusd -C
ExecReload=/bin/kill -HUP $MAINPID

[Install]
WantedBy=multi-user.target
```

```
# configure file /etc/raddb/user
vi /usr/lib/systemd/system/user
```

```
user1 Cleartext-Password := "testing"
        Reply-Message := "Hello, %{User-Name}"
steve   Cleartext-Password := "testing"
        Service-Type = Framed-User,
        Framed-Protocol = PPP,
        Framed-IP-Address = 172.16.3.33,
        Framed-IP-Netmask = 255.255.255.0,
        Framed-Routing = Broadcast-Listen,
        Framed-Filter-Id = "std.ppp",
        Framed-MTU = 1500,
        Framed-Compression = Van-Jacobsen-TCP-IP
```

```
# configure file /etc/raddb/radiusd.conf
vi /usr/lib/systemd/system/radiusd.conf

prefix = /
exec_prefix = ${prefix}
sysconfdir = ${prefix}/etc
localstatedir = ${prefix}/var
sbindir = ${exec_prefix}/sbin
logdir = ${localstatedir}/log/radius
raddbdir = ${sysconfdir}/raddb
radacctdir = ${logdir}/radacct

name = radiusd

confdir = ${raddbdir}
modconfdir = ${confdir}/mods-config
certdir = ${confdir}/certs
cadir   = ${confdir}/certs
run_dir = ${localstatedir}/run/${name}
db_dir = ${raddbdir}
libdir = ${exec_prefix}/lib
pidfile = ${run_dir}/${name}.pid
correct_escapes = true
max_request_time = 30
cleanup_delay = 5
max_requests = 16384
hostname_lookups = no
```

```
# configure file /etc/raddb/sites-available/default
vi /etc/raddb/sites-available/default

    listen {
        ipaddr = *
        port = 0
        type = acct
    }

    authenticate {
        Auth-Type PAP {
                pap
        }
        Auth-Type CHAP {
                chap
        }
        Auth-Type MS-CHAP {
                    mschap
            }
        mschap
        digest
        eap

    }
```

MySQL configuration

```
 # initialize the Mysql
 mysql
 MariaDB [(none)]> SET password for 'root'@'localhost'=password('123456';
 MariaDB[(none)]> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%'IDENTIFIED BY
'123456' WITH GRANT OPTION;
 MariaDB [(none)]> flush privileges;
 # establish the radius database
 mysql -uroot -p123456
 MariaDB [(none)]> create database radius;
```

```
 # import the radius table structure
 mysql -u root -p123456 radius < /etc/raddb/mods-
config/sql/main/mysql/schema.sql
 # verify everything works
 MariaDB [(none)]> show databases;
 MariaDB [(none)]> use radius;
 MariaDB [radius]> show tables;
 MariaDB [radius]> exit
```

```
 # configure mysql authentication in freeradius
 cd /etc/raddb/mods-enabled
 ln -s ../mods-available/sql

 # modify mysql in Radius
 vi /etc/raddb/mods-available/sql

 driver = “rlm_sql_mysql”
 dialect = “mysql”
 # insert data in mysql
 mysql -uroot -p123456
 use radius;
 insert into radgroupreply (groupname,attribute,op,value) values ('user','Auth-
Type',':=','Local');
 insert into radgroupreply (groupname,attribute,op,value) values
('user','Service-Type',':=','Framed-User');
 insert into radgroupreply (groupname,attribute,op,value) values
('user','Framed-IP-Address',':=','255.255.255.255');
 insert into radgroupreply (groupname,attribute,op,value) values
('user','Framed-IP-Netmask',':=','255.255.255.0');
 # we make a new account where username=BNSS6, Password=123456
 insert into radcheck (username,attribute,op,value) values ('BNSS6','Cleartext-
Password',':=','123456');
 insert into radusergroup (username,groupname) values ('test','user');
```

```
 #configure the radius client information
 vim /etc/raddb/client.conf
 #because the router have connected to the VPN
 #VPN server address:192.168.0.43
 #All EAP verify request should come from internal network.
 client private-network-1 {
        ipaddr          = 192.168.0.43/24
        secret          = BNSS6
 }
```

```
 #start the radiusd
 radiusd -X
```

firewall configure to enable the radius and MySQL service

```
systemctl start firewalld.service
systemctl enable firewalld.service
firewall-cmd --zone=public --add-port=3306/tcp --permanent
firewall-cmd --zone=public --add-port=1812/tcp --permanent
firewall-cmd --zone=public --add-port=1812/udp --permanent
firewall-cmd --reload
```

Router configuration

We want to deploy Radius Client in router and make a gateway to gateway VPN connection.
The router has been flashed to OpenWrt in advance. If you haven't done this, refer to https://openwrt.org/downloads

Equip it in the router

```
opkg update
#because the wapd-mini doesn't support for WPA2-EAP
opkg remove wpad-mini
opkg install wpad
opkg install eapol-test-openssl
```

```
vim /etc/config/wireless/wifi-iface

config wifi-iface 'default_radio0'
    option device 'radio0'
    option network 'lan'
    option mode 'ap'
    option ssid 'BNSS6_Radius'
    option encryption 'wpa2'
    option server '192.168.0.65'
    option key 'BNSS6'
```

OpenVpn Client in router

```
vim /root/vpn/config
#configure the OpenVPN
client
dev tun
proto udp
remote IP 1194
resolv-retry infinite
nobind
user nobody
group nogroup
persist-key
persist-tun
ca /etc/openvpn/ca.crt
cert /etc/openvpn/client3.crt
key /etc/openvpn/client3.key
tls-auth /etc/openvpn/ta.key 1
cipher AES-256-GCM
comp-lzo
verb 3
mute 20
mssfix 1400
```

Then get `ca.crt` `client3.crt` `client3.key` `ta.key` from VPN server and move them to `/etc/openvpn/`.

```
 #start the vpn
/etc/init.d/openvpn restart
#to read the log
logread -f
```

Configure Firewall to Allow Forward

```
iptables -I FORWARD -o tun0 -j ACCEPT
iptables -t nat -I POSTROUTING -s 0.0.0.0/0 -d 0.0.0.0/0 -o tun0 -j MASQUERADE
```

now the devices which connect to the routers can access to internal network.

1.

##Google authenticator in Apache Server
CentOS7

1. download in server and run

```
sudo apt-get install apache2-prefork-dev
sudo apxs2 -i -a -n authn_google mod_authn_google.so
google-authenticator
```

2. move and change the configuration

```
mv .google_authenticator /etc/apache2/ga_auth
#configure authn_google.conf
vim /etc/apache2/mods-available/authn_google.conf

Options FollowSymLinks Indexes ExecCGI
AllowOverride All
Order deny,allow
Allow from all
AuthType Basic
AuthName "Secret"
AuthBasicProvider "google_authenticator"
Require valid-user
GoogleAuthUserPath ga_auth
GoogleAuthCookieLife 3600
GoogleAuthEntryWindow 2
```

```
#start  mod_authn_google
sudo a2enmod authn_google && sudo service apache2 restart
```

3. add authentication user

```
sudo apt-get install libpam-google-authenticator
google-authenticator
```

4. scan the QR code from mobile devices
5. change permission

```
sudo chmod 640 BNSS6 && sudo chown root:www-data BNSS6
```

Snort IDS

1. Install the required prerequisites with the following command.

```
sudo yum install -y gcc flex bison zlib libpcap pcre libdnet tcpdump
```

```
sudo yum install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
sudo yum install -y libnghttp2
```

2. Install Snort with yum.

```
sudo yum install
http://www6.atomicorp.com/channels/atomic/centos/7/x86_64/RPMS/snort-2.9.6.1-1.el7.art.x86_64.rpm
```

3. Update the shared libraries using the command underneath

```
sudo ldconfig
```

4. Set up username and folder structure

```
sudo groupadd snort
sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort
```

```
sudo mkdir -p /etc/snort/rules
sudo mkdir /var/log/snort
sudo mkdir /usr/local/lib/snort_dynamicrules
```

```
sudo chmod -R 5775 /etc/snort
sudo chmod -R 5775 /var/log/snort
sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules
sudo chown -R snort:snort /etc/snort
sudo chown -R snort:snort /var/log/snort
sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules
```

```
sudo touch /etc/snort/rules/white_list.rules
sudo touch /etc/snort/rules/black_list.rules
sudo touch /etc/snort/rules/local.rules
```

5. Configuring the network and rule sets

```
sudo vi /etc/snort/snort.conf
```

```
# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.0.43/32
# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET !$HOME_NET
# Path to your rules files (this can be a relative path)
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules
```

```
# Set the absolute path appropriately
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules
# unified2
# Recommended for most installs
output unified2: filename snort.log, limit 128
```

```
include $RULE_PATH/local.rules
```

6. Validating settings

```
ln -s /usr/lib64/libdnet.so.1.0.1 /usr/lib64/libdnet.1
```

```
sudo snort -T -c /etc/snort/snort.conf
```

7. Test the configuration

```
sudo vi /etc/snort/rules/local.rules
```

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001;
rev:001;)
```

```
sudo snort -A console -i eth0 -u snort -g snort -c  /etc/snort/snort.conf
```

Complete the following command by pressing TAB, and you can see the log of the alert.

```
snort -r /var/log/snort/snort.log.
```

Firewall

The firewall software we choose to use is Firewalld. It builds onto the iptabls and you can think of it as a 'front end' of iptables.

Our default zone/policy is public/drop. Only the service or protocol we explicitly enabled can go through the firewall.

We will take the firewall configuration on the OpenVPN gateway and the authentication Apache proxy as examples because they are the most vulnerable points in our network.

Please Note these regulations in Firewalld:

- Only incoming packets are processed.
- Packets are always processed in top to down direction (rich rules are on top of services or ports).
- Once a packet matches with rule, associate action (allow or deny) will be taken immediately for that packet.
- Packet will not be available for further processing.

Configurations

On VPN Gateway

```
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: dhcpv6-client openvpn ssh
  ports: 1194/tcp
  protocols:
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule protocol value="icmp" drop
        rule family="ipv4" service name="openvpn" log prefix="OpenVPNLimit"
level="warning" limit value="20/m" accept limit value="20/m"
        rule family="ipv4" source address="192.168.0.0/24" service name="ssh"
log prefix="SSH Access" level="notice"
        rule port port="1194" protocol="tcp" log prefix="OpenVPNLimit"
level="warning" limit value="20/m" accept limit value="20/m"
```

On Authentication Apache proxy

```
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: dhcpv6-client https http ssh
  ports: 444/tcp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
        rule family="ipv4" source address="192.168.0.0/24" service name="ssh"
log prefix="SSH_Limit" level="warning" limit value="1/m" accept limit
value="2/m"
        rule port port="443" protocol="tcp" log prefix="HttpsLimit"
level="warning" limit value="100/s" accept limit value="100/s"
        rule port port="444" protocol="tcp" log prefix="HttpsLimit"
level="warning" limit value="100/s" accept limit value="100/s"
        rule port port="80" protocol="tcp" log prefix="HttpsLimit"
level="warning" limit value="100/s" accept limit value="100/s"
```

General Setup Command

1. Install and Enable Your Firewall to Start at Boot

```
sudo yum install firewalld
sudo systemctl enable firewalld
sudo reboot
```

2. Explore the default

```
sudo firewall-cmd --list-all
```

```
#output
public (default, active)
  target: default
  icmp-block-inversion: no
  interfaces: eth0 eth1
  sources:
  services: ssh dhcpv6-client
  ports:
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

3. Add services and ports to the zone

```
sudo firewall-cmd --zone=public --permanent --add-service=http
sudo firewall-cmd --zone=public --permanent --add-service=https
sudo firewall-cmd --zone=public --permanent --add-port=444/tcp
```

4. Add rich rules

```
sudo firewall-cmd --permanent --add-rich-rule='rule protocol value=icmp
drop'
sudo firewall-cmd --permanent --add-rich-rule='rule port port="443"
protocol="tcp" accept limit value="100/s" log prefix="HttpsLimit"
level="warning" limit value="100/s"'
sudo firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source
address=192.168.0.0/24 service name=ssh log prefix="SSH Access"
level="notice" log limit value=5/m prefix="Too Much SSH Tried"
level="warning" accept'

```

5. Reload the firewall

```
sudo firewall-cmd --reload
```

Usage

Access from London Office

If you are in ACME London Office, the only thing you need to do to access our internal resources, is to connect to our London Office Router with WPA2 Authentication. We assume in this case, all the computer is pre-configured by engineers from ACME.

Access from your Home

In this case, you need to do some configurations.

Connect to our VPN

We use OpenVPN as our VPN implementation. To connect our VPN, you need to download OpenVPN Connector which is available here https://openvpn.net/download-open-vpn/. We will provide you with a `.ovpn` file, which you can import to OpenVPN Connector and connect.

We provide two methods to authenticate you to our VPN server, the digital certificate or username/password with Google Authenticator.

Using Certificate

For the digital certificate method, the public `cert` and private `key` are embedded in the `.ovpn` file, which means you can directly connect to our VPN server without further configuration.

Using Username/Password with 2FA

For 2FA method, you need bind your Google Authenticator with your account in advance. When you try to connect to our VPN server, you need use `password | code in google-authenticator` as the right password to log in.

Authenticate to our Web/File Server

You are in our internal network now. But you can't access our Web/File Server before further authentication.

Still, there are two ways for you to authenticate, Certificate or Kerberos.

Using Kerberos

We assume you are using Windows platform.
Four steps are required before accessing our internal service

1. Download and install Firefox web browser (We currently don't support Chrome or IE).
    1. Navigate to the URL `about:config`.
    2. Click "Accept the Risk and Continue".
    3. Type `negotiate-auth` into the filter at the top of the page, in order to remove most of the irrelevant settings from the list.
    4. Double-click on `network.negotiate-auth.trusted-uris` and enter

       ```
       bnss6.com
       ```

    5. Find `network.auth.use-sspi` option and change it to False.
2. Download MIT Kerberos for windows from http://web.mit.edu/kerberos/dist/.

   1. Follow the instructions in the installer to complete the installation process.
   2. Change to C:\ProgramData\MIT\Kerberos5 directory. Replace `krb5.ini` file with the code below and save it.

      ```
      [libdefaults]
      dns_lookup_realm = false
      ticket_lifetime = 24h
      renew_lifetime = 7d
      forwardable = true
      rdns = false
      default_realm = BNSS6.COM

      [realms]
      BNSS6.COM = {
              kdc = krb.bnss6.com:88
              admin_server = krb.bnss6.com:749
              default_domain = BNSS6.COM
      }

      [domain_realm]
      .bnss6.com = BNSS6.COM
      bnss6.com = BNSS6.COM
      ```

3. Configure `hosts` mapping. We haven't deployed our DNS server yet :(
   Add 2 entries into `hosts` file located at C:\Windows\System32\drivers\etc. You need Administrator privilege to edit this file.

   ```
   192.168.0.91  krb.bnss6.com
   192.168.0.162  web.bnss6.com
   ```

4. Get tickets from our Kerberos Server.
   1. Open your MIT Kerberos client
   2. Click Get Ticket on the up right corner. A dialogue box for entering username and password should appear.
   3. Use the username and password below for testing.

      ```
      windows@BNSS6.COM
      windows
      ```

   4. You should have a granted ticket now.

Using Certificate

If you want to use this way to authenticate, you should have a key/cert pair.

There are multiple ways to utilize the certificate. Our suggestion is use `openssl` to convert your key/cert pair to `.p12` format certificate. You can simply import this file to your web browser and the authentication will be done automatically.

Now open Firefox and enter `web.bnss6.com`!

Use your Phone to exchange file

You will need a Android phone and VPN connection.

Download `OpenVPN` app and use exactly the same config file you used in Windows PC.

Then download `Seafile` app and it will prompt you to enter information. If ACME requires a higher security level, you may have to log in with advanced configurations. More details in the demo.

Enable Two Factor Authenticator

1. download google authenticator in smart phone
2. generate QR code from Seafile Setting
3. use mobile device scan the QR code generated from Seafile to bind.