Extending a Debian OpenVPN server with Multi Factor Authentication via Google Authenticator

This document describes how to extend an existing OpenVPN server configuration with MFA. There are multiple documents about this on the Internet but I found that I still had to tweak some things and the explanation of the client was wanting.

For inspiration of this document, I happily stole from the following Internet documents:
https://vorkbaard.nl/how-to-set-up-openvpn-with-google-authenticator-on-pfsense/
https://medium.com/we-have-all-been-there/using-google-authenticator-mfa-with-openvpn-on-ubuntu-16-04-774e4acc2852

1. Extending the Server daemon

The server is running Debian 9 on a nuc. We'll first create a complete backup of the /etc directory for contingency and then install a few new packages.

```
# tar -cvf backup-etc-27-08-2019-without-mfa.tar etc
# apt-get install -y libqrencode3 libpam-google-authenticator
```

Now add a user and group "gauth" that will query the Google Authentication service. It will use a configuration directory /etc/openvpn/google-authenticator:

```
# addgroup gauth
# useradd -g gauth gauth
# mkdir /etc/openvpn/google-authenticator
# chown gauth:gauth /etc/openvpn/google-authenticator
# chmod 0700 /etc/openvpn/google-authenticator
```

From now on, OpenVPN users will authenticate using their Linux/UNIX credentials besides the client certificate.
This implies that EVERY OpenVPN client will from now on need to be created in Linux as well…
Once the user has been created, they will require a MFA secret key etc. This is stored by default in the $HOME directory under .google-authenticator. This is the default.

Another option is to create the users, but to store the users' google authenticator information at an alternative location. This is what I personally prefer, hence the /etc/openvpn/google-authenticator is created. For each client a file will be created with the client – name that contains the MFA credentials.

Extend the openvpn.conf server file (sometimes referred to as server.conf file) with the plugin:

```
plugin /usr/lib/openvpn/openvpn-plugin-auth-pam.so openvpn
```

And of course, a ***restart of the server*** daemon is in order.

Here comes the painful part.. the PAM configuration. PAM has always been tricky to fiddle with and you should be careful configuring it as it is possible to lock yourself out..

Create a new PAM configuration file /etc/pam.d/openvpn. First find out where your PAM library is to be found as the location seems to vary widely..:

```
# find / -name  pam_google_authenticator.so
/lib/x86_64-linux-gnu/security/pam_google_authenticator.so
```

And create the PAM configuration file for OpenVPN:

```
# vi /etc/pam.d/openvpn
auth requisite /lib/x86_64-linux-gnu/security/pam_google_authenticator.so
secret=/etc/openvpn/google-authenticator/${USER} user=gauth forward_pass
```

Explanation: for authentication purposes, the requisite is that the google authenticator shared object verifies that user’s authentication according to the file provided with the “secret” keyword. So a different file will be provided for each individual user. The query to Google is run as user “gauth”.

As I understand it, the Google credentials are parsed once the regular Linux password has been verified by the other PAM modules.

You are now finished with the Server setup. That wasn’t too hard.. but we’re not out of the woods yet..

2. Creating the user credentials for MFA

In this document I presume you already have a user setup for OpenVPN so I’ll skip the certificate setup.
What we’ll have to do now, is to add the user to Linux/UNIX. As the user’s credentials are the only thing we are interested in (shadow database), we will not allow the OpenVPN user to login :

```
# useradd -d /home/testuser -s /bin/false testuser
# passwd testuser
```

For the new user we will generate the MFA information:

```
# su -c "google-authenticator -t -d -r3 -R30 -f -l 'OpenVPN Server' -s
/etc/openvpn/google-authenticator/testuser" - gauth
```

What this does is: as user “gauth” you run the google-authenticator command and save the output under `/etc/openvpn/google-authenticator/testuser.`

What happens is a bit startling.. a QR code is generated in the terminal window! It looks roughly like this:

And a few questions come up:

```
By default, tokens are good for 30 seconds. 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. If you experience problems with
poor time synchronization, you can increase the window from its default
size of +-1min (window size of 3) to about +-4min (window size of
17 acceptable tokens).
Do you want to do so? (y/n)  y
```

This will create the configuration file /etc/openvpn/google-authenticator/testuser that contains something like this:

```
/etc/openvpn# more google-authenticator/testuser
************* ****
" RATE_LIMIT 3 30
" WINDOW_SIZE 17
" DISALLOW_REUSE
" TOTP_AUTH
********
********
```

These are emergency credentials in case you use MFA for eg SSH authentication and you can no longer login. If you do not have access to the QR code, you can eg manually configure the App by using the first security code in this file.

3. Setting up the App for your OpenVPN server

What you do now with the QR code is as follow:
In the Google Authenticator AP, **you SCAN THE QR code**. It seems a bit strange.. but works like a charm. It will add an entry for your OpenVPN server in the APP.

4. Creating the OpenVPN client configuration file

There is quite a bit of information that can be found in the client .ovpn file: symmetric keys, private keys, loads of keywords etc etc.
In older days, the .ovpn file would refer to certificate files etc. These days, you can combine all the necessary files into a single .ovpn file, which makes it very easy to distribute to eg tablets.

It is easiest to use a script to generate the .ovpn file. In the .ovpn file, 2 new statements are added:

```
ns-cert-type server // this might change in future, see the end of this doc
auth-user-pass
```

If you use a template for all your clients that contains the keywords, (you probably do) then these two lines will have to be added to that template.

The remains of the file are the same. A nice script can be found on the Internet that creates the .ovpn file as well the Google MFA information and emails the credentials.

It can be found here:
https://gist.github.com/egonbraun/7176976fe05ece092410462facf0adb6

Note: <u>**do NOT blindly implement this script**</u>, check your easy-rsa setup properly prior to using this script.
The nice thing is that the script will email the entire file with credentials to the user, using the CLI "**`mutt`**" command. To test "**`mutt`**" first, try the following:

```
# echo "Here is your OpenVPN client configuration" | mutt -s "Your OpenVPN
configuration" -a "/etc/hosts" -- "testuser@ulimit.nl"
```

For some reason this does returns an error message "**`GPGME: CMS protocol not available`**" -> google it to disable gpg. Even with the message, it does work however.

5. Does it work?....

Once the new .ovpn file has been imported into the client and the Google Authenticator is setup, make sure you authenticate as follows:
Username: Linux/UNIX username
Password: <Linux/UNIX password><6-digit Google Authenticator code>

Example: if you Linux/UNIX password is test123 and your Google Authenticator code is 123456, then your password for the VPN is: test123123456

This is Linux so it probably won't work the first time. There are loads of reasons why this might go wrong. So first restart your OpenVPN server deamon and monitor both the logfiles /var/log/auth.log (for PAM related messages) and /etc/openvpn/openvpn.log. That latter filename might be called differently in your case.

With a bit of luck.. you will find that this professional way of authenticating your users does work and it looks pretty fancy. And is secure. ;)

6. Very important: troubleshooting

The QR code that you receive when creating the MFA credentials, are personal to the user. So what that means is that if as an administrator, you want to verify his/her account that is not simply possible by using your MFA code with Google Authenticator!! You will need to use the QR code for that individual account within Google Authenticator and scan it..

Without the QR code, you can still re-create the Google Authenticator entry by using the contents of the file /etc/openvpn/google-authenticator/{USER}.

7. The OpenVPN iPhone client..

Once you are ecstatic about this above.. your laptop will probably work as OpenVPN client. BUT.. the iPhone won't ,- or at least not in my case. This might be due to loads of reasons.. in my case the standard OpenVPN client at least provided decent logging. It was complaining about the "ns-cert-type" keyword mentioned above.

Solution: replace "**`ns-cert-type`**" with "**`remote-cert-tls server`**" and Bob's your uncle.

8. Replication

Using VRRP (keepalived) it's relatively simple to implement a redundant OpenVPN solution as the OpenVPN files can simply be synced from one server to the next.
Can you replicate the Google Authenticator credentials as well?
The simple answer is YES. All you have to do is sync the **`/etc/openvpn/google-authenticator/{USER}`** files. ;)

9. What if I don't have access to the QR code or I lose my Google Authenticator device?

If you want to configure the Google Authenticator App without using the QR code, then simply use the security code from **`/etc/openvpn/google-authenticator/{USER}`** and enter the code manually in the App.
It is the first line in the file and looks roughly like this: **`CBTX7AFS3QZEFPPWR3PNPIQIQ4`**