OpenVPN Server with 2FA

Table of contents:

1. Introduction

(This file is kept in PDF format on github due to issues with images in large LibreOffice files. For the original version, please go to https://docs.google.com/document/d/1Oi7P6lfvIGav1RmK-4oA6MnTkRxUDBK10LZE6a8CwAc/edit?usp=sharing)

OpenVPN is an open-source VPN solution that provides VPN functionality based on SSL and PKI infrastructure, although authentication by username/password is also supported, but only for users. Server always authenticates itself by a certificate issued by a CA trusted by connecting clients. As an additional security feature, a two-factor user authentication can be added. The package can be installed on Windows or Linux machine and operate in a server or client mode. In this document we describe the process of installation and configuration of OpenVPN service with 2FA using Google Authenticator on an Ubuntu 18.04 server. The goal of this project is to create a gateway that provides access to servers that are located in the local network behind it.

2. Installation of a package.

OpenVPN can be installed from official repository:

```
sudo apt-get install openvpn easy-rsa
```

The installer sets up a virtual interface (tun) drivers that acts as a tunnel endpoint in a virtual network and creates folder structure in **/etc/openvpn**. By default **Easy-RSA**, an open-source CA solution, is also installed there, and we will use it to issue certificates to OpenVPN server in our environment.

If not, execute the following:

sudo cp -r /usr/share/easy-rsa /etc/openvpn

3. Service configuration

3.1. Configure PKI

The configuration of the server part of OpenVPN can be divided in two stages: setting up the PKI infrastructure and edition of OpenVPN config file.
In brief, to configure a CA using Easy-RSA we need to create a CA, whose certificate will be used to issue certificates to server and then issue a certificate for the server. Also Easy-RSA is used to generate a file with Diffie-Hellman group parameters for traffic encryption. It all can be done with bat-scripts provided with Easy-RSA. They are based on variables configured in **vars** file that defines certificates parameters like Country code, City, Organisation, etc. After these variables are set we can proceed to PKI creation:

- Run command shell with root credentials:
  ```
  sudo bash
  ```
- Jump to the EasyRSA folder:
  ```
  cd /etc/openvpn/easy-rsa
  ```
- Run the following command to load variables required for EasyRSA scripts operation:
  ```
  source ./vars
  ```
- Run the following command to prepare files and folders for the new PKI:
  ```
  ./clean-all
  ```
  This command deletes all the files generated previously, use with caution!

- Run the following command to create CA:
  ```
  ./build-ca
  ```
- Run the following command to generate Diffie-Hellman group file:
  ```
  ./build-dh
  ```

- Run the following command to issue the server certificate:
  ```
  ./build-key-server <server-name>
  ```
- Run the following command to issue the clients certificate:
  ```
  ./build-key <client-name>
  ```
  The same certificate will be used for all clients
- To issue additional certificates in the future don't forget to run `source ./vars` before creating other certificates**.**

Important note: all variables in **vars** file must be defined (there must be no any empty fields after **=** sign)!

Here are the variables defined for our environment:
```
export KEY_COUNTRY="US"
export KEY_PROVINCE="AZ"
export KEY_CITY="Tucson"
export KEY_ORG="GBCInc"
export KEY_EMAIL="techsupport@onlinegbc.com"
export KEY_CN=onlinegbc.com
export KEY_ALTNAMES=funauto.com
export KEY_NAME=GBCInc
export KEY_OU=FUNAuto
```

At this point we'll get a certification authority sufficient to serve our OpenVPN infrastructure. The next step is to configure the VPN service.

3.2. OpenVPN configuration basic configuration

For additional security another HMAC signature key is generated to authenticate the TLS hand-shaking process. This key must be configured for usage in both clients and server config file. Only connection attempts from clients who has the right signature cert are processed. Use the following command to generate the key:

```
openvpn --genkey --secret ta.key
```

This file must be put to **/etc/openvpn** folder to be readable by openVPN process. Adapted settings in the sample config are enough for the service to function. By default OpenVPN working directory is the one where its config file is located, that is, **/etc/openvpn**. We need to copy the certificates created by EasyRSA there. Here's the list of key options set in our case:
`port 1194` - *port the service listens (it's a default value)*
`proto udp`
`dev tun` - *type of the Tunnel interface (tun is recommended for most cases)*
`ca ca.crt - CA certificate`
`cert <server-name>.crt` - *server certificate public key*

`key <server-name>.key` - *server certificate private key*
`dh dh2048.pem`
`topology subnet` - *VPN topology: a subnet that all participants' Tun interfaces belong to*
`server 10.8.0.0 255.255.255.0` - *VPN network address range. 10.8.0.1 is the server IP*
`push "route 10.0.0.0 255.255.255.0"` *- add a route to the on-premise network on clients*
`duplicate-cn` - *allow usage of the same certificate by many clients*
`tls-auth ta.key 0` - *use the TLS authentication and mark this side as server*
`cipher AES-256-CBC` - *use AES-256 encryption (must match on server and clients)*
`status /var/log/openvpn/openvpn-status.log` - *status log file (connected clients, IPs, etc.)*
`log /var/log/openvpn/openvpn.log` - *log file (wipes every time server is restarted)*
`reneg-sec 0` - *disable re-authentication requests (must match on server and clients)*

3.3. System configuration

Since the server is used as a gateway to access servers in the local subnet, some system parameters must be adapted. First of all, the OpenVPN service must be started on system boot:

```
    sudo systemctl enable openvpn
```

Then, inter-interface packet forwarding must be enabled. To do that the following parameter must be modified in **/etc/sysctl.conf**:

`net.ipv4.ip_forward=1` - *uncomment this variable to enable packet forwarding*

This modification will take effect after network service restart and will be implemented every time the server restarts. To apply the setting immediately, run the following command:

```
    sudo sysctl -w net.ipv4.ip_forward=1
```

Now a NAT must be implemented to perform address translation for all packets going from OpenVPN clients to the local network. The source address for these packets is changed to the server's **enp1s0** interface IP. This will make these packets appear as they are originated from the server itself. It allows to avoid reconfiguration of the routing on the servers that must communicate with VPN clients. Run the following command to enable NAT:

```
    sudo iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j
MASQUERADE
```

But it only affects the running server. The NAT rule won't be applied after server's restart. To make this setting persistent the following packets must be installed:

```
sudo apt-get install iptables-persistent netfilter-persistent
```

After installation finishes issue the following command to save the iptables configuration:

```
sudo netfilter-persistent save
```

4. Two-Factor Authentication

Google Authenticator is chosen as an authentication provider for 2FA. It's an open-source solution with applications available for Android and Apple platforms. The applications generate a security code valid for 30 seconds. During this time span a client must connect to the server and pass their username with password and security code.

4.1. Installation on server side

To install the software required for Google Authenticator to work run the following command:

```
apt install libqrencode3 libpam-google-authenticator
```

We create a separate user that will have exclusive access to a folder where user tokens are kept:

```
addgroup grp-gauth
useradd -g usr-gauth grp-gauth
passwd usr-gauth
```

Create a folder for tokens and modify access rules:

```
mkdir /etc/openvpn/google-authenticator
chown usr-gauth:grp-gauth /etc/openvpn/google-authenticator
chmod 0700 /etc/openvpn/google-authenticator
```

To tell OpenVPN to use Google Authenticator the following line should be added to the server.conf (in single line):

```
plugin /usr/lib/x86_64-linux-gnu/openvpn/plugins/openvpn-plugin-auth-pam.so
"openvpn login USERNAME password PASSWORD"
```

This line instructs OpenVPN to use **PAM Authentication plugin** and pass username and password of connecting clients as parameters.
Then the authentication parameters file should be created in **/etc/pam.d folder:**

```
touch /etc/pam.d/openvpn
```

This is the contents of the file (in single line):

```
auth required -/lib/x86_64-linux-gnu/security/pam_google_authenticator.so
secret=/etc/openvpn/google-authenticator/${USER} user=usr-gauth forward_pass
```

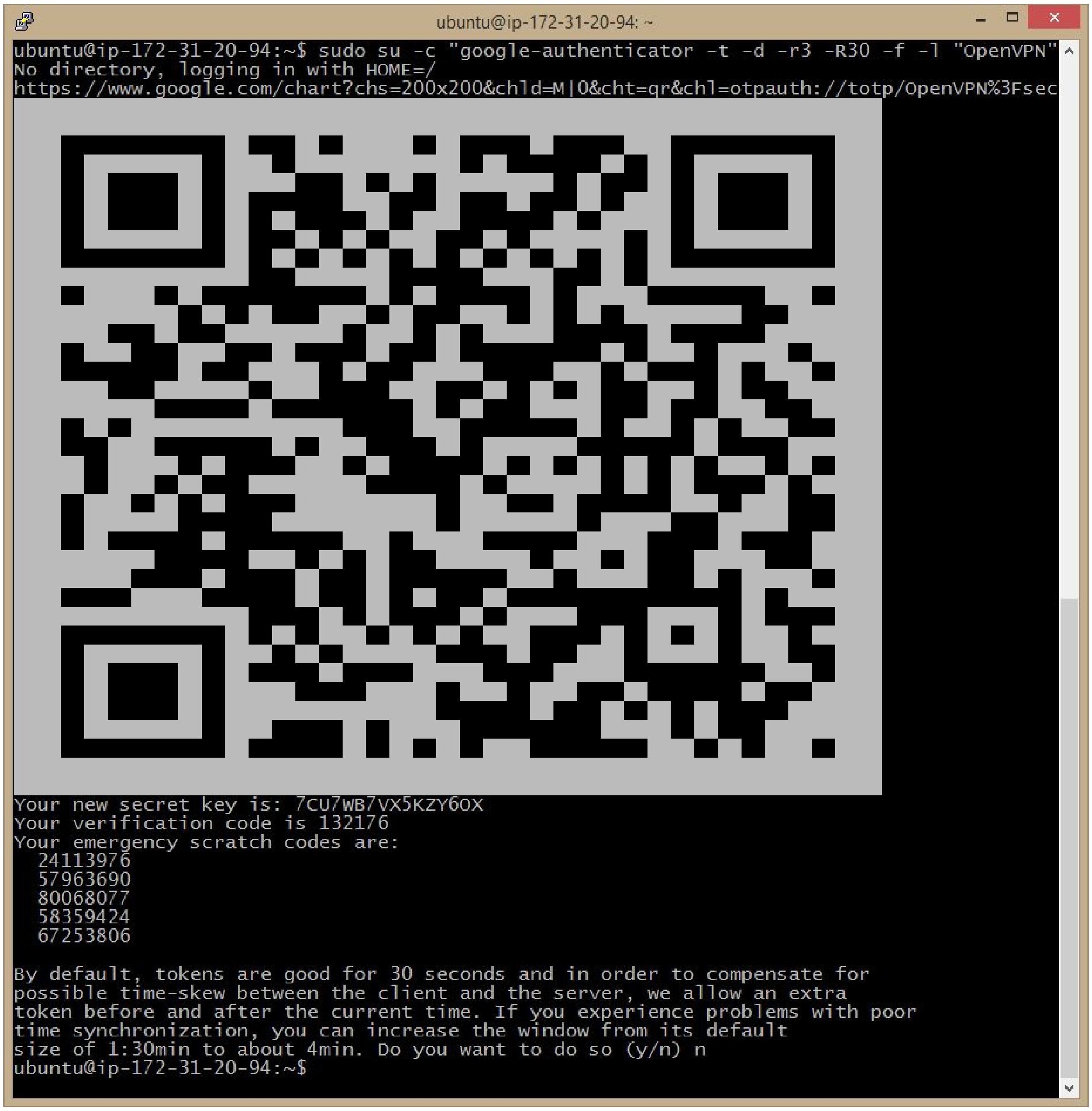
4.2. Adding a new OpenVPN user

Since users are authenticated by their username and password in our setup, aside from the clients certificate which is the same for all users, a database of users logins should be kept somewhere. The easiest way is to use system users DB that is stored in /**etc/passwd** file.To create a new user run the following commands on the VPN server:

useradd <username> - *create a user account*
passwd <username> - *set the password for the new user*
sudo su – usr-gauth

Enter the following as a single line:
google-authenticator -t -d -r3 -R30 -f -l ‘OpenVPN’ -s /etc/openvpn/google-authenticator/<username>

That’s how the output of successful token creation looks like:



```
ubuntu@ip-172-31-20-94:~$ sudo su -c "google-authenticator -t -d -r3 -R30 -f -l "OpenVPN"
No directory, logging in with HOME=/
https://www.google.com/chart?chs=200x200&chld=M|0&cht=qr&chl=otpauth://totp/OpenVPN%3Fsec

Your new secret key is: 7CU7WB7VX5KZY6OX
Your verification code is 132176
Your emergency scratch codes are:
  24113976
  57963690
  80068077
  58359424
  67253806

By default, tokens are good for 30 seconds and 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 1:30min to about 4min. Do you want to do so (y/n) n
ubuntu@ip-172-31-20-94:~$
```

And here’s a brief explanation for the **google-authenticator** command:

- `su -c "<command>" - usr-gauth` - *The* `<command>` *is being run in context of* ***usr-gauth*** *user because he has the necessary rights to* ***/etc/openvpn/google-authenticator*** *folder*
- `google-authenticator` - *the script that generates tokens*

- `-t` - *use time-based authentication*
- `-d` - *disallow reuse of the same code*
- `-r3 R30` - *limit logins to 3 per every 30 seconds*
- `-f` - *write token key and settings without first confirming with user*
- `-s` - *location where the token file for the* `<username>` *will be put. We use* ***/etc/openvpn/google-authenticator****.*

In the output the following information is important:

- The **QR code** that a user can scan with the application to add a new token
- The **secret key** line that contains the ID that a user can add in the application manually. On the screenshot above it's `7CU7WB7VX5KZY6OX`.

The token ID must be passed to the user so that they could add it on the Google Authenticator app installed on their smartphone.
 - To connect to the OpenVPN server the user must provide their login name, their password and a one-time code generated by Google Authenticator. Since the feature of passing the code to the server isn't supported by OpenVPN clients, the code can be included in the password field following the password. For instance, if the user's password is "ovpn-pass" and the code is "123456", then the user must put "ovpn-pass123456" in the password field.
On the router we made a port-forwarding rule that will forward all the packets destined to its external address (70.167.120.50) and UDP port 25109 to the internal address of the server (192.168.1.110), UDP port 1194. UDP port 25109 was chosen to avoid the usage of the default OpenVPN port as a security measure.

5. Client Setup.

5.1. Configuration file.

The configuration file has the same format for any system and contains the client-specific options. Typically, OpenVPN client programs for Windows, Android and IOS automatically recognise configuration files with **.ovpn** extension. Below is the key parameters for OpenVPN in client mode:
`client` - *instruct OpenVPN to run in client mode*
`dev tun` - *same as with server config*
`proto udp` - *same as with server config*
`remote <Server public IP> 1194` - *server IP or DNS name and port for connection*
`ca ca.crt` - *the path is relative because searching for certificates in user's OpenVPN folder is hardcoded*
`cert client-certificate.crt` - *client certificate public key*
`key client-certificate.key` - *client certificate private key*
`auth-user-pass` - *use username/password authentication method*
`tls-auth ta.key 1` - *use the TLS authentication and mark this side as client*

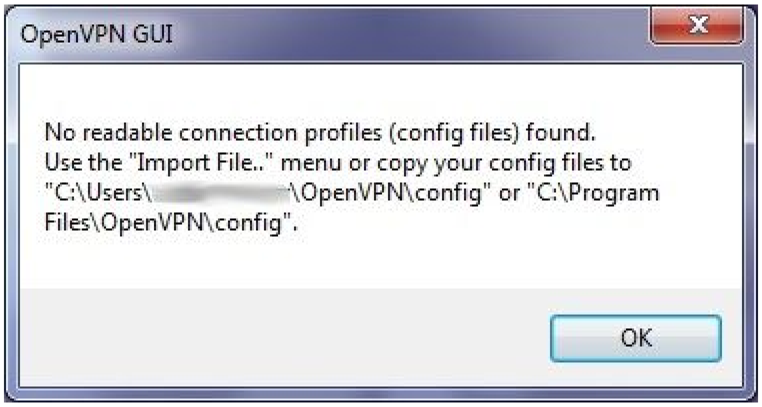
`cipher AES-256-CBC` - *use AES-256 encryption (must match on server and clients)*
`reneg-sec 0` - *disable re-authentication requests (must match on server and clients)*

The file must be saved **.ovpn** extension and shared with users.

5.2. OpenVPN GUI for Windows (Installation and Configuration)

There's a client software for different OSes available for downloading on the OpenVPN official web site: https://openvpn.net/community-downloads/. Below is the description of the configuration process for Windows clients.

Installation procedure is pretty straight-forward and doesn't require any changes in components selection proposed by the installer. When starting up, OpenVPN looks for the config file and required certificates in **%username%/OpenVPN/config/client** folder. If it doesn't exist OpenVPN will show a message like this:



No readable connection profiles (config files) found.
Use the "Import File.." menu or copy your config files to "C:\Users\[redacted]\OpenVPN\config" or "C:\Program Files\OpenVPN\config".

and create it. This can happen if OpenVPN GUI starts for the first time. Several configs and certificates can be placed there allowing to connect to different servers.

5.3. Linux CLI (Installation and Configuration)

There's no difference in packages for server or client mode: the mode OpenVPN runs in is defined by the config file. So, to run OpenVPN on any Linux distributive, it can be installed using a package manager for a particular distro. For example, on Ubuntu it can be installed with the following command:

```
sudo apt-get install openvpn
```

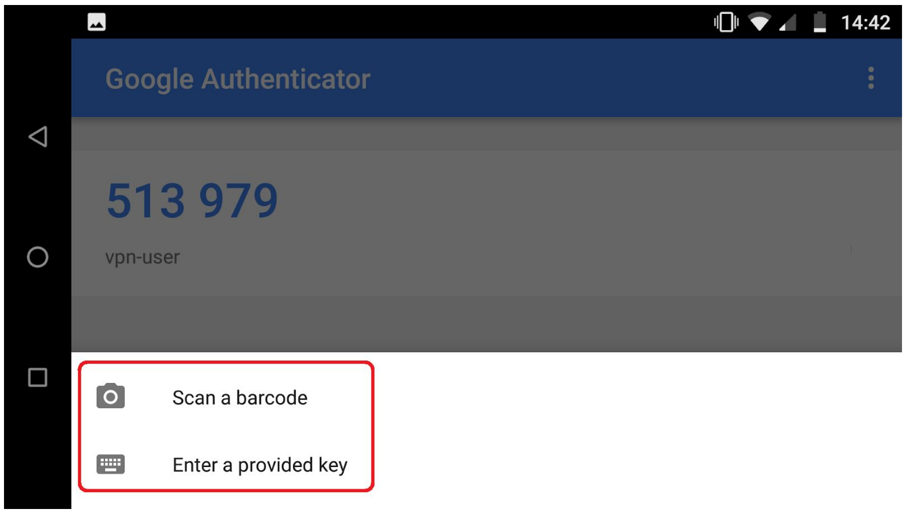
The following files must be placed together with the configuration file used in the command above:

- **ca.crt** - Root CA certificate's public key
- **client-certificate.crt** - client certificate's public key

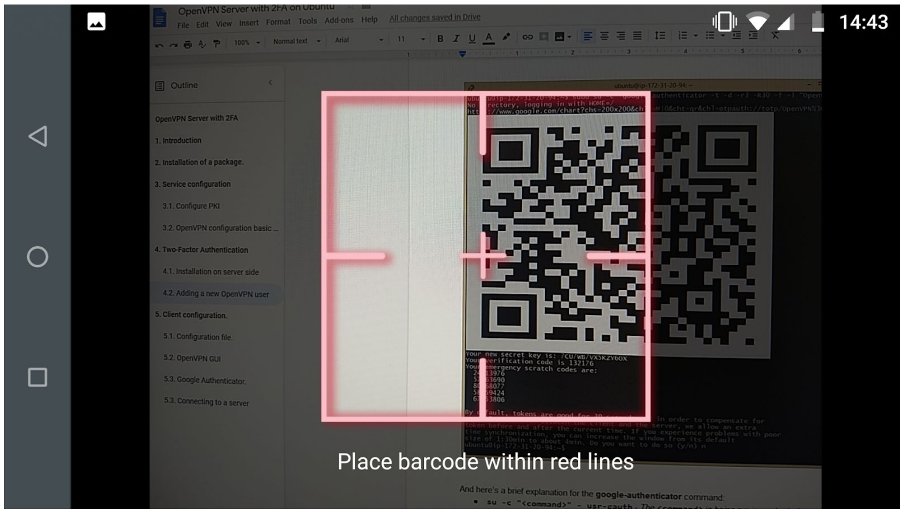
- **client-certificate.key** - client certificate's private key
- **ta.key** - TLS Auth key

5.4. Google Authenticator/LastPass Authenticator.

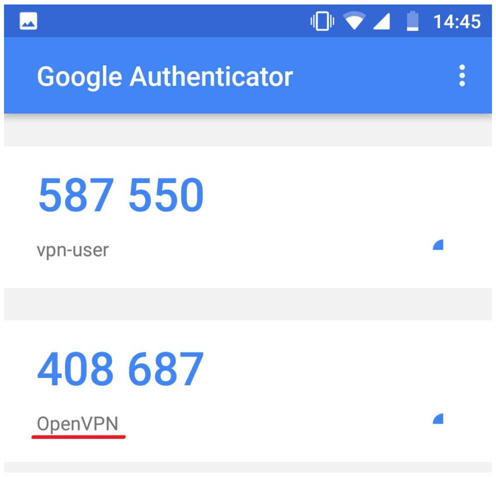
The client must install the **Google Authenticator** app on their smartphone (or a similar application like **LastPass Authenticator** which is transferable across devices) to be able to get one-time codes for 2FA. Below are the instructions for Google Authenticator app.
After installation a user token must be added to it to generate the security codes. Press the "**+**" in lower right corner of the screen and select "**Scan a barcode**" to read the QR code generated on the server by **google-authenticator** command or "**Enter a provided key**" to add the token ID manually:



To scan the code place it within the red frame:



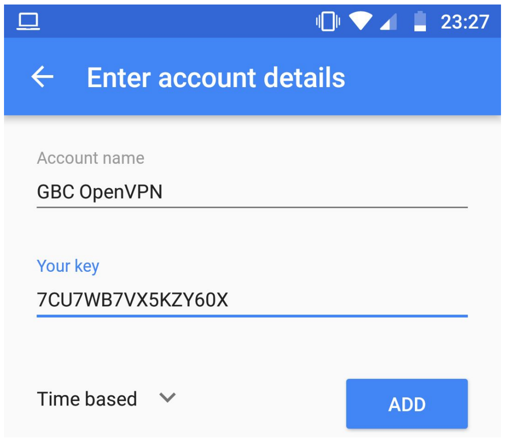
It will be recognised and added automatically (note the label for the newly added token, it's the field defined with `-l` parameter during token creation with **google-authenticator**):



587 550
vpn-user

408 687
OpenVPN

If the token ID is added manually, enter the account label, the ID, select Time-based as authentication type and tap Add button:

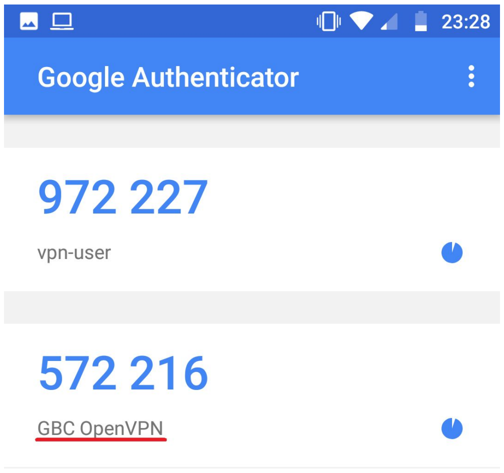


Account name
GBC OpenVPN

Your key
7CU7WB7VX5KZY60X

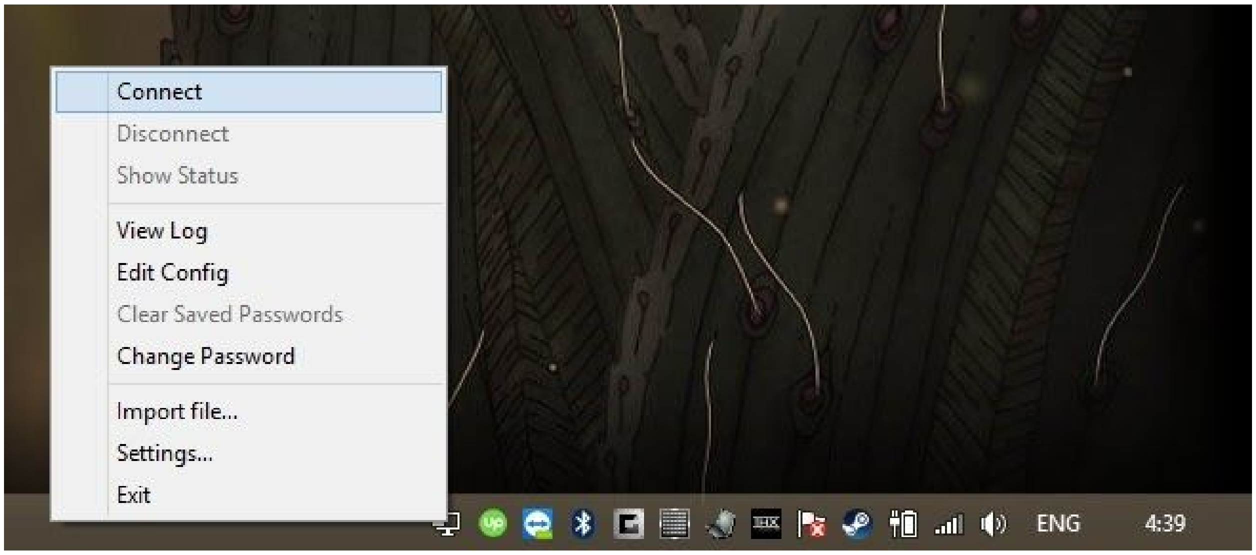
Time based

The newly added account will appear on the main screen:

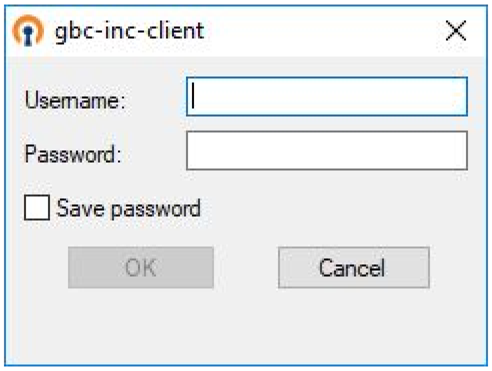


5.5. Connecting to a server with Windows client

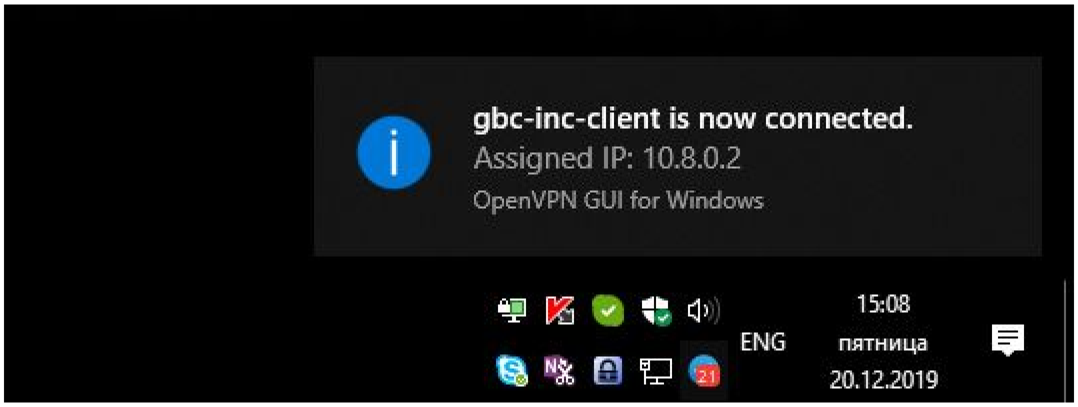
To connect to a server manually run the OpenVPN GUI, right click on the icon in the System tray and click **Connect**:



A credentials request window appears:



The user must provide their login name in the **Username** field and their password and a one-time code generated by Google Authenticator in the **Password** field. Since there's no separate field for the code, it can be included in the Password field following the password. For instance, if the user's password is "*ovpn-pass*" and the code is "*123456*", then the user must put "*ovpn-pass123456*" in the **Password** field.
Wait for the message balloon about successful connection pops up. This would mean that VPN tunnel has been established:



gbc-inc-client is now connected.
Assigned IP: 10.8.0.2
OpenVPN GUI for Windows

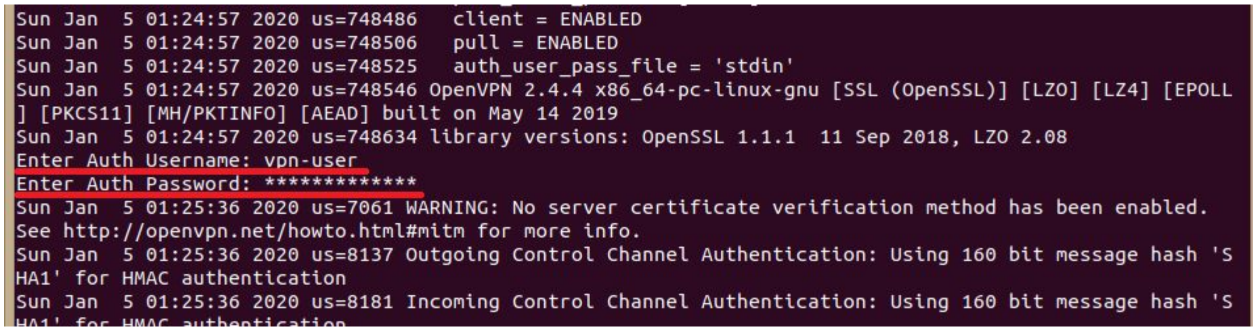
Now we can reach the servers in the local network address range by their local IP addresses.

5.6. Connecting to a server with Linux CLI

To use the Linux CLI a **Terminal** application should be run. It exists in one form or another on any Linux. The connection to the server can be initiated with the following command:

```
sudo openvpn --cd </path to config file> --config client.ovpn
```

`--cd` option orders openvpn to change the working directory to the one where all the necessary files located.
The username and password are requested during the negotiation:



```
Sun Jan  5 01:24:57 2020 us=748486   client = ENABLED
Sun Jan  5 01:24:57 2020 us=748506   pull = ENABLED
Sun Jan  5 01:24:57 2020 us=748525   auth_user_pass_file = 'stdin'
Sun Jan  5 01:24:57 2020 us=748546 OpenVPN 2.4.4 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL
] [PKCS11] [MH/PKTINFO] [AEAD] built on May 14 2019
Sun Jan  5 01:24:57 2020 us=748634 library versions: OpenSSL 1.1.1  11 Sep 2018, LZO 2.08
Enter Auth Username: vpn-user
Enter Auth Password: *************
Sun Jan  5 01:25:36 2020 us=7061 WARNING: No server certificate verification method has been enabled.
See http://openvpn.net/howto.html#mitm for more info.
Sun Jan  5 01:25:36 2020 us=8137 Outgoing Control Channel Authentication: Using 160 bit message hash 'S
HA1' for HMAC authentication
Sun Jan  5 01:25:36 2020 us=8181 Incoming Control Channel Authentication: Using 160 bit message hash 'S
```

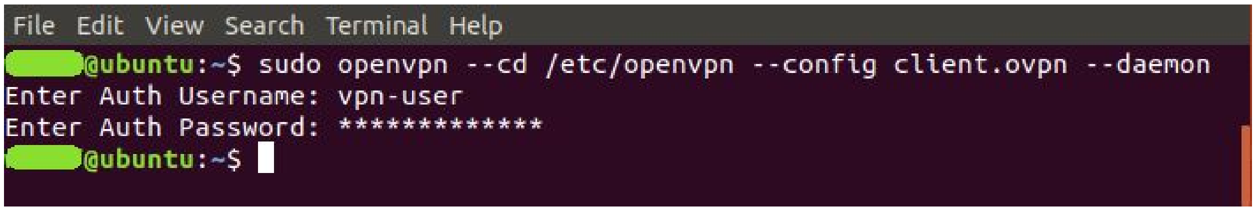
Again - the one-time code generated by Google Authenticator should be entered following the password.
`Initialization Sequence Completed` - this is the message that tells that the connection was successful. After that the Terminal window can be minimized and the OpenVPN will print out some logging information there.

If the Terminal window is closed, the OpenVPN connection is terminated.

OpenVPN can be run with the `--daemon` option:

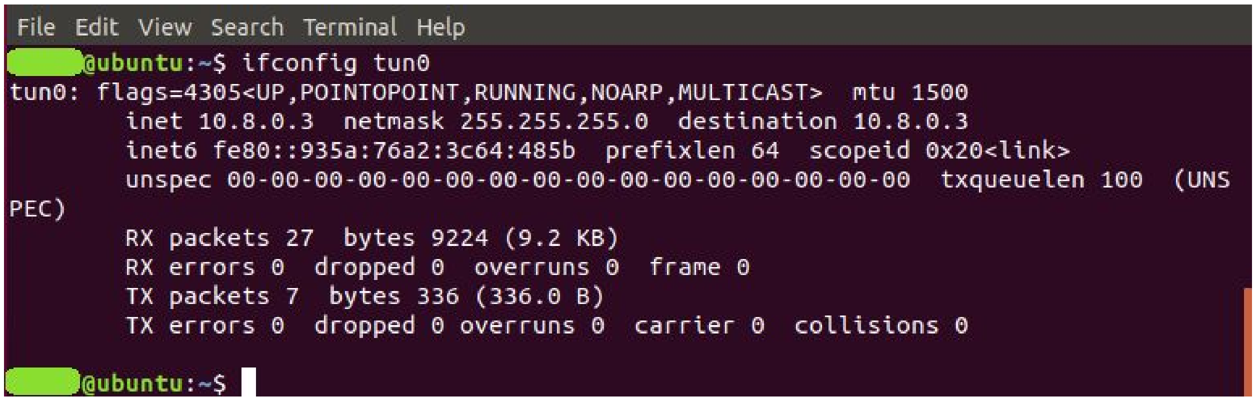
```
    sudo openvpn --cd </path to config file> --config client.ovpn
--daemon
```

This will make it run in the background. It means that there will be no output from OpenVPN and the Terminal window can be closed after the connection:



```
[redacted]@ubuntu:~$ sudo openvpn --cd /etc/openvpn --config client.ovpn --daemon
Enter Auth Username: vpn-user
Enter Auth Password: *************
[redacted]@ubuntu:~$
```

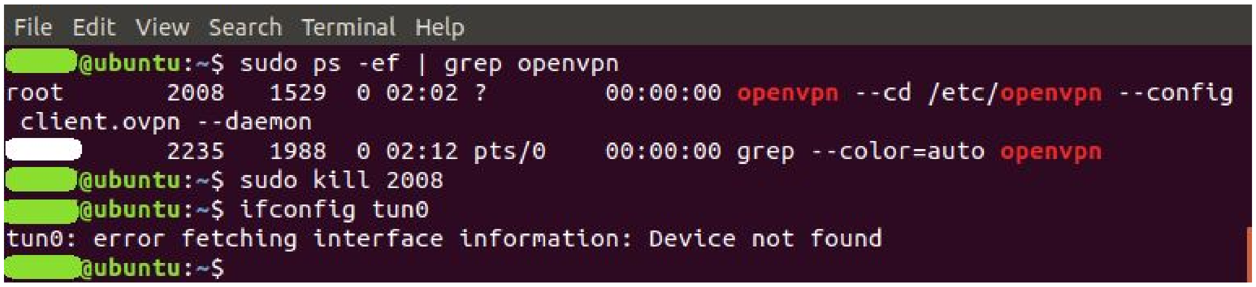
The connection is successful if a **tun0** interface appeared in the system:



```
[redacted]@ubuntu:~$ ifconfig tun0
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.8.0.3  netmask 255.255.255.0  destination 10.8.0.3
        inet6 fe80::935a:76a2:3c64:485b  prefixlen 64  scopeid 0x20<link>
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 100  (UNS
PEC)
        RX packets 27  bytes 9224 (9.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7  bytes 336 (336.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[redacted]@ubuntu:~$
```

To drop the connection the OpenVPN process should be identified and terminated using **`ps`** and **`kill`** commands:



```
[redacted]@ubuntu:~$ sudo ps -ef | grep openvpn
root       2008   1529  0 02:02 ?        00:00:00 openvpn --cd /etc/openvpn --config
 client.ovpn --daemon
[redacted]       2235   1988  0 02:12 pts/0    00:00:00 grep --color=auto openvpn
[redacted]@ubuntu:~$ sudo kill 2008
[redacted]@ubuntu:~$ ifconfig tun0
tun0: error fetching interface information: Device not found
[redacted]@ubuntu:~$
```

Troubleshooting

1. If VPN does not work after some apparently unrelated changes or upgrade, try this:
    I. Confirm that the OpenVPN client is running and the target is reachable from inside the intranet
    Ii. sudo systemctl stop openvpn
    Iii. If needed, try a reboot of the VPN server
2. After a move from Physical to VM, the network did not route properly. Some of the possible fixes were these commands (root as root or sudo):
    I. iptables -L
    Ii. iptables -t nat -D 1
    Iii. iptables -t nat -D POSTROUTING 1
    Iv. iptables -t nat -L
    V. iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o enp0s3 -j MASQUERADE
    Vi. netfilter-persistent save
3. You may get an error "/dev/net/tun: no such file or directory" while starting the client
    In such a case, have InterServer support enable tun on the underlying server. (ref ticket # 670200 (PAU-614-98584)
    Run these steps:
    cd /dev
    sudo mkdir net
    cd net
    sudo mknod tun c 10 200
    sudo chmod 666 tun
    After this, try starting the openvpn client again, or reference this:
    https://serverfault.com/questions/456953/trying-to-set-up-openvpn-server-on-a-vps

Appendix

A. Reference client configuration file for GBC OpenVPN

Below is the working config file for connection to GBC OpenVPN Server (mandatory parameters are `highlighted`)

```
################################################################
# Save the contents of this file as client.ovpn                #
# Put it in %username%/OpenVPN/config/client folder on Windows #
# Put it in /etc/openvpn folder on Linux                       #
################################################################


# Disable the data channel TLS renegotiation
# on the server and push this setting to a client
reneg-sec 0

# Specify that we are a client and that we
# will be pulling certain config file directives
# from the server.
client

# Use the same setting as you are using on
# the server.
# On most systems, the VPN will not function
# unless you partially or fully disable
# the firewall for the TUN/TAP interface.
;dev tap
dev tun

# Windows needs the TAP-Win32 adapter name
# from the Network Connections panel
# if you have more than one.  On XP SP2,
# you may need to disable the firewall
# for the TAP adapter.
;dev-node MyTap

# Are we connecting to a TCP or
# UDP server?  Use the same setting as
# on the server.
;proto tcp
```

```
proto udp

# The hostname/IP and port of the server.
# You can have multiple remote entries
# to load balance between the servers.
remote 69.244.22.119 1194
;remote 69.242.227.235 1194
;remote my-server-2 1194

# Choose a random host from the remote
# list for load-balancing.  Otherwise
# try hosts in the order specified.
;remote-random

# Keep trying indefinitely to resolve the
# host name of the OpenVPN server.  Very useful
# on machines which are not permanently connected
# to the internet such as laptops.
resolv-retry infinite

# Most clients don't need to bind to
# a specific local port number.
nobind

# Downgrade privileges after initialization (non-Windows only)
;user nobody
;group nogroup

# Try to preserve some state across restarts.
persist-key
persist-tun

# If you are connecting through an
# HTTP proxy to reach the actual OpenVPN
# server, put the proxy server/IP and
# port number here.  See the man page
# if your proxy server requires
# authentication.
;http-proxy-retry # retry on connection failures
;http-proxy [proxy server] [proxy port #]

# Wireless networks often produce a lot
# of duplicate packets.  Set this flag
# to silence duplicate packet warnings.
;mute-replay-warnings

# SSL/TLS parms.
# See the server config file for more
# description.  It's best to use
# a separate .crt/.key file pair
# for each client.  A single ca
# file can be used for all clients.
ca ca.crt
cert client-certificate.crt
key client-certificate.key

# Enable username/password auth plugin.
auth-user-pass

# Verify server certificate by checking that the
# certicate has the correct key usage set.
# This is an important precaution to protect against
# a potential attack discussed here:
#  http://openvpn.net/howto.html#mitm
#
# To use this feature, you will need to generate
# your server certificates with the keyUsage set to
#   digitalSignature, keyEncipherment
# and the extendedKeyUsage to
#   serverAuth
# EasyRSA can do this for you.
remote-cert-tls server

# If a tls-auth key is used on the server
# then every client must also have the key.
tls-auth ta.key 1

# Select a cryptographic cipher.
# If the cipher option is used on the server
# then you must also specify it here.
# Note that v2.4 client/server will automatically
# negotiate AES-256-GCM in TLS mode.
# See also the ncp-cipher option in the manpage
cipher AES-256-CBC

# Enable compression on the VPN link.
# Don't enable this unless it is also
# enabled in the server config file.
#comp-lzo

# Set log file verbosity.
verb 3

# Silence repeating messages
;mute 20
```