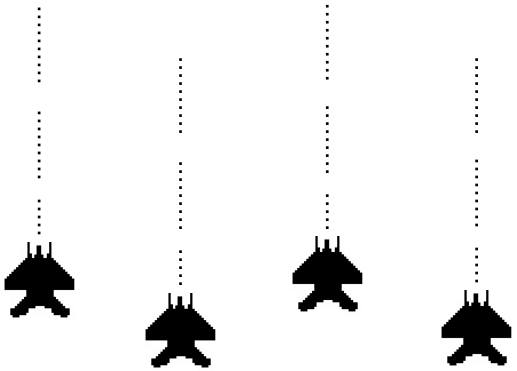
TROUBLE SHOOTING openVPN



HOWTO INSTALL AND RUN
A SELF-HOSTED VPN SERVICE
WITH OPENVPN

this manual is part of the VPN zines collection from psaroskalazines.gr

it is made with hand drawn icons by the author and layout design is done with scribus

cover and colophon font:
www.fontspace.com/category/atari

content font:
www.1001fonts.com/white-rabbit-font.html

many thanks to:
digitaldefenders.org for the financial support
systerserver.net for hosting the git repository of the files

find the author on mastodon
mara@systerserver.town

INTRODUCTION

This zine navigates the reader through an openVPN installation and troubleshooting.
It is a manual meant for network admins of all levels who want to provide vpn services on the premises (aka from their self-hosted, autonomous servers), as well as for users who want to experiment with networking, privacy and security.
The content involves a terminal and command line (aka Command Line Interface - CLI), and focuses on Linux systems. It is written in an accessible language by an author who came to the networking universe at the crispy (late) age of 26 and grasps what it means to be an adult newbie in the competitive milieu of computer geeks and hackers.

INDEX

INVENTORY

server: the machine on which openVPN is running in server mode, and forwards traffic from users to the internet or private networks, such as a virtual private server (VPS), or a bare metal server.

client: the device from which the user connects to the internet via the openVPN server, such as laptops and mobile phones.

WAN and LAN: Wide/Local Area Network. the second called also private network and is inaccessible from the internet.

router: the device that forwards data from one network to another. It has built-in ethernet ports, and home routers usually have a Wi-Fi access point and an internal modem to pass traffic along a cable or DSL line. The traffic traversing a router is formatted in the TCP/IP protocol, and its routing rules control what traffic can pass.

gateway: A more general term than the above, for devices or machines that forward and receive packets to the Internet. It can be a cable or dsl modem, a router, or a server setup as a firewall.

pc: personal computer
IANA: aka Internet Assigned Numbers Authority, which looks after global IP address allocation

COMMANDS

ifconfig: shows our devices' IP address
ip addr: successor to ifconfig
nslookup: finds the IP of a domain name.
traceroute: tracks the packets from an IP network on their way to a given host.
EOF: End of File
echo "some text" >> filename: adds some text at EOF
cat filename1 >> filename2: adds content of file at EOF
routel: list device's routes in pretty output format
ping: checks the connection to a server.
e.x ping systerserver.net outputs:

```
64 bytes from systerserver.net (xx.xx.xx.xx): icmp_seq=1 ttl=47 time=97.3 ms
64 bytes from systerserver.net (xx.xx.xx.xx): icmp_seq=2 ttl=47 time=95.6 ms
```

which ensures that the site is reachable.

tail or less: for reading logs, e.x
sudo tail -n 100 /var/log/syslog, option -n 100 shows the last 100 lines of the file
vim or emacs or nano: text editor for configuration files, the latter is more an intuitive.
Note: On a remote server accessible only via a terminal (SSH) we need one of the above terminal based editors. on a pc we can use any text editor as long as we ensure that no random characters are introduced, e.x in some windows' editors the ^M character get inserted at the end of a line.

<cmd>, </cmd>: command block
#: comment on the command

DOWNLOAD & INSTALL

1. openvpn (version 2. x)
2. easy-rsa (if not installed. together with openvpn)

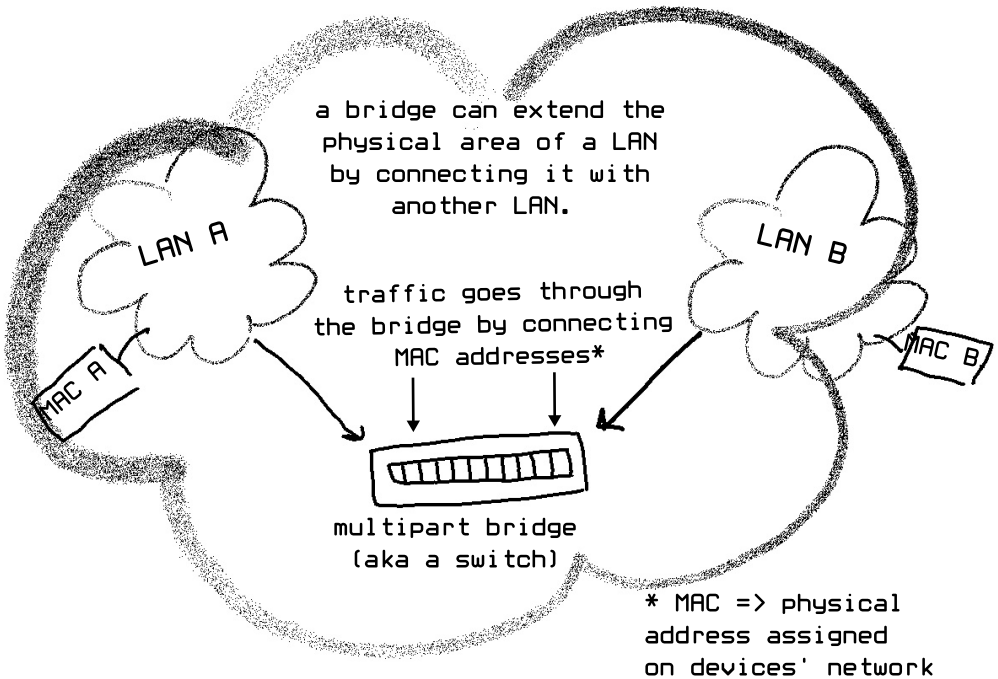
Some linux flavors need openssl, lzo and pam dependencies. For other operating systems visit the openvpn's docs online (see useful links at the CHEAT SHEET chapter).

Depending on our linux system, we can install with yum, apt-get, or emerge.
Here we continue with a debian/ubuntu example
<cmd> sudo apt install openvpn </cmd>

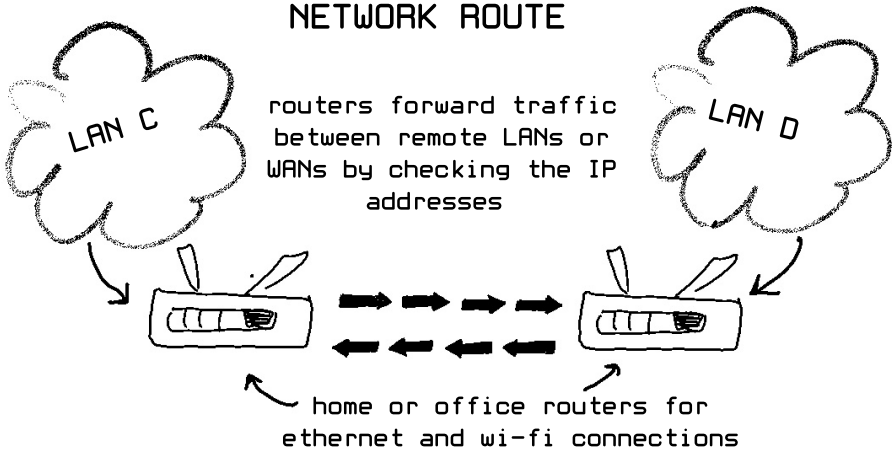
Note 1: In the OpenVPN settings, we need to select between a routed or a bridged type of network. In most cases, when we need to connect securely over the internet or circumvent websites' censorship, we need a routed VPN, not bridged. See schema next page for the differencies.

Note 2: Then we need to choose a privte subnet for the openvpn tunnel. Available subnets from IANA are: 10. 0. 0.0/8, 172.16. 0.0/12, 192.168. 0.0/16. In this manual we will be using the first.

NETWORK BRIDGE



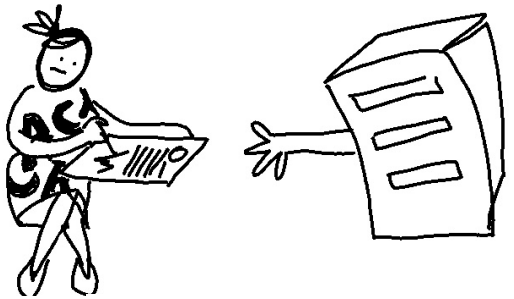
NETWORK ROUTE



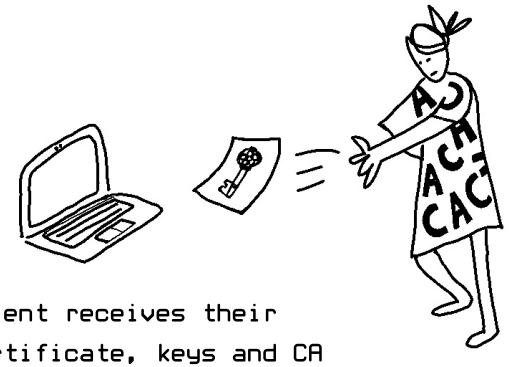
GENERATE CERTIFICATES

We are setting up a VPN with an X 509 PKI (aka Public Key Infrastructure, see next page schema), which establishes an authentication method with certificates and keys. easy-rsa software helps us to create these files.
If it came bundled with openvpn, then it should be at "/etc/openvpn/easy-rsa". Not there? Fetch it with:
<cmd>
sudo apt install easy-rsa
locate easy-rsa | grep README
less <path/to/README>
</cmd>
The README explains how to create a symbolic link to the source scripts of easy-rsa, so we can use them with openvpn.

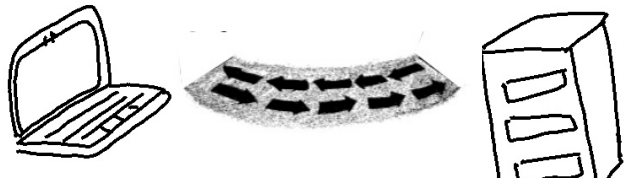
We can see many dirs and files inside easy-rsa! We start by editing the vars file to set variables and thus save time later while we generate the certificates.
<cmd> vi vars #or nano </cmd>
The "vars" file concerns the Certificate Authority, which issues and signs the server and clients' certificates. So we give names related to our team, location, etc. Also we set "KEY-SIZE = 2048" or higher, e.x 4096. Once we have saved & exited the vars file, we source the variables, to make the terminal remember the values we just set:
cmd> . ./vars </cmd>
To remove old or unused certificates:
<cmd> ./clean-all </cmd>
Note: above command will delete all existing keys.



CA signes server and client's certificates



Client receives their certificate, keys and CA certificate from the openvpn server admin.



Client and server will negociate tunnel connection with tls authentication and the Diffie Helleman key

CA & SERVER & CLIENT KEYS

To build keys for the Certificate Authority (aka CA):
<cmd> . /build-ca </cmd>
hit "enter" for the questions, and for the last 2 questions hit "yes".
At the end, check keys were generated under the directory "/etc/openvpn/easy-rsa/keys".

To build keys for the server:
<cmd> . /build-key-server <name-of-our-server> </cmd>
Hit enter for almost all questions except:
"common NAME" for which we can give for example the server's name or our group/organisation name.
And for the following two last questions enter "y":
1."Sign the certificate?"
2. "1 out of 1 certificate-requests certified, commit?"

To build keys for the user (aka client):
<cmd> ./build-key <name-of-user-or-group> </cmd>
Repeat for each user.

To build the Diffie Helman parameters:
<cmd> ./build-dh </cmd>

Note: In this manual we will be using "server" as the name of our server, so the generated keys will be respectively server.crt and server.key. Likewise we will be using "client" as the name of our user/group. We can give whatever names we want, just we need to note the paths of those created keys.

SERVER CONFIGURATIONS

Next we copy the file "/etc/openvpn/sample-conf" to "/etc/openvpn/server.conf" and open it for editing the settings.

Note:";" at the beginning of the line signifies the directive is commented out (has no effect). To uncomment it and make it effective we delete the ";".

General notes for editing the server.conf:
We set a random port which is not in use by other services. E.x port 25 is taken by SMTP, 443 for HTTPS, and 22 for SSH. We usually need a routed VPN so we uncomment the "dev tun" directive, see note 1 in "DOWNLOAD & INSTALL" chapter. A udp protocol is preferable to tcp for faster connections. Then we give the path to the ca, cert and key files:
ca ca.crt
cert server.crt
key server.key

For the Diffie hellman setting we give the path to the diffie hellman pem, which we generated just before.
Supply a tunnel subnet for ex.
"Server 10.4.0.0 255.255.255.0"
see note 2 in "DOWNLOAD & INSTALL".

CONFIGURATIONS TIPS

If we supply group certificates (same cert for a bunch of users) we need to uncomment "duplicate-cn". Also we can uncomment the keepalive directive.

It's good practice to generate the extra key for HMAC (hash message authentication) as it is indicated in the server.conf file.
tls-auth ta.key 0
mode server
tls-server

For the encryption layer, good options are:
cipher AES-256-CBC, auth SHA256, and key-direction 1.

Also we uncomment the vpn-link compression directive, uncomment the nobody user and nobody group directive, as well as the persist key and tunnel directives.

We add a path for the log files, and finally we set the verbose mode to "verb 3", except if we want to debug as we will see later.
Very useful is to enable the certificate revocation check for when we want to disable a user's access. We can do this in the end of the server.conf:
crl-verify crl.pem
See link reference at CHEAT SHEET.

CLIENT CONFIGURATIONS

The client configuration file is similar to the server's with the correct relative or absolute paths to the CA file (ca.crt), the client's keys, and the extra HMAC related key with the directive"tls-auth ta.key 1"
Also we need to add the remote IP or domain name of the VPN server and the port we have set in the server.conf:
remote <public ip> 1196

All above 4 files have to be sent to the users who are going to connect to the VPN service. These files can be all bundled in the client.conf file with the use of tags.
In this case, we comment out their paths in the client.conf
; ca ca.crt
; cert client.crt
; key client.key
Instead we do
<ca>
CA cert here
</ca>
<cert>
Client's cert here
</cert>
<key>
client's key here
</key>
<tls-auth>
HMAC key here
</tls-auth>

RUN THE SERVICE

Once openvpn is installed we can run it across reboots on a linux server and client by editing /etc/default/openvpn and remove the "#" in front of "AUTOSTART=all".
Then with the system unit scripts we can run:
<cmd>
sudo systemctl enable openvpn-server@<name>.service
sudo systemctl daemon-reload
sudo systemctl restart openvpn-server@<name>.service
#for the client machine it's openvpn-client@<conf>.service
</cmd>

Explanation:
the system unit scripts for openvpn are located in /lib/systemd/system/. So e.x for running openvpn service on the client, the unit script is "openvpn-client@.service". Whatever <name> we put between "openvpn-client@" and ".service" gets passed inside this unit script, which executes the following:
Goes to the directory /etc/openvpn and runs "openvpn --config <name>.conf".
So it's good practice to put the client.conf under the /etc/openvpn directory. Or change the paths of client.conf in the unit script (as a sudo user).

Did it work? No? Let's see how to troubleshoot:
Common problem is the connection cannot be established between the server & the client.

SERVER SIDE SHOOTS

* check openvpn on server is running:
<cmd> sudo systemctl status openvpn-server@server </cmd>
The output gives error hints. We can read horizontally across the output with the left/right arrow keys.
If it isn't running, we start the service with:
<cmd> sudo systemctl start openvpn-server@server </cmd>

* check logs and look for error messages (log file path is set in the server.conf):
<cmd> tail-n 100 /var/log/openvpn </cmd>
If logs are located elsewhere we replace the path respectively.
If we want to navigate the whole file and keep new messages appending to the log:
<cmd> less +F /var/log/openvpn </cmd>
If we want to navigate inside the log file and stop new messages appending, we type "ctr + C".
With capital F we start appending messages again at the end of file. We exit with "ctr+C" and then "q".

*Check if the tunnel IP exists with:
<cmd> sudo ifconfig tun0 #for Ubuntu </cmd>
<cmd> sudo ip add show dev tun0 #Debian </cmd>

* ping the client's tunnel IP, which can be found from the client machine with the above commands:
ifconfig or ip addr

CLIENT SIDE SHOOTS

* inside the client.conf search for "verbosity" and increase it to 6 or more and run openvpn again. Don't forget to set it back to 3 or lower when vpn connection is fixed, otherwise the logs will consume too much disk space.

* then run openvpn directly, not as a service, and check the output logs in the terminal for error hints.
<cmd>sudo openvpn --config client.conf </cmd>
Stops with typing "ctr+C".

* make sure client's configuration has the same settings as the server in the relevant directives. Errors in terminal output indicates which settings need correction.

* ping the server's tunnel IP to see if it's reachable from the client's pc. This can be derived from the tunnel subnet we set in server.conf from which the server takes the 1st address. In our example it is 10.4.0.1

ROUTING TABLE CHECK

*Look at the routing table in the client's pc with the commands "route" or "routel". Check if server's tunnel IP is under target/destination column, and client's tunnel IP is under source column. ex. if client's tunnel IP is 10.4.0.10 and server's 10.4.0.1, it should look like:

target	gateway	source	proto	scope	dev
default	192.xx.xx.xx		static		wlp2s0
10.4.0.1	10.4.0.9				tun0
10.4.0.9		10.4.0.10	kernel	link	tun0

In case we have connected another server/client at the other side of the tunnel it will appear as a peer with a tun0 IP, e.x 10.4.0.9. We see how to add a peer (a private server ora restricted website) in the next chapter.

ACCESS RESTRICTED SITES

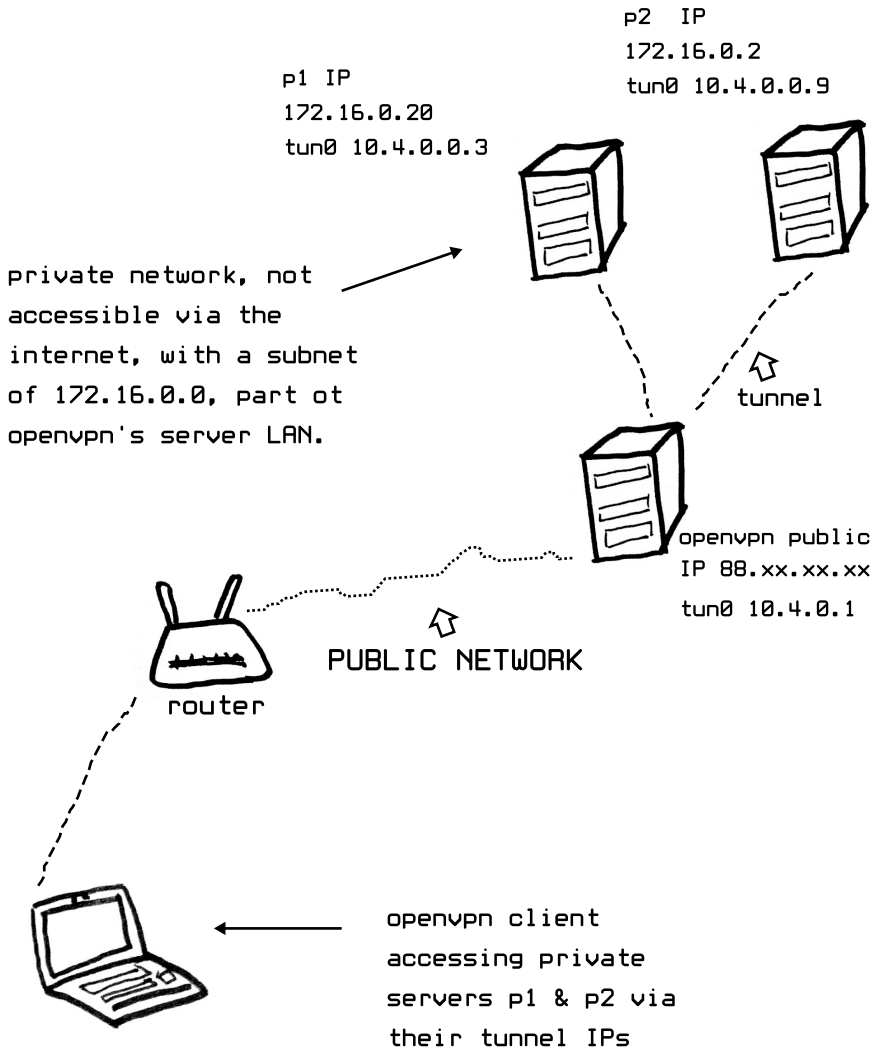
Once openvpn is running on both the server & the client, we would want to access private servers behind the openvpn server, or access restricted websites. We can also forward all network traffic through the openvpn but it will make our web traffic slow and even unstable if client's pc is behind a dynamic IP, see cheat sheet at the end of this zine for extra resources on this.

Here we go through an example of how to access a geolocally restricted website or a private server from the LAN of the openvpn server. To do so we need to add in the server.conf the website we want to reach via openvpn or the private IP or subnet of the server.

Let's say youtube is restricted in our location, so we need to add in the server.conf the following:
push "route youtube.com 255.255.255.255"
If we want to access a private server, we add it's IP or its private subnet:
push "172.16.0.0 255.255.0.0"

Note 1: In this case the openvpn server is also a network gateway.
Note 2: We basically add the private subnet of the openvpn server.

PRIVATE VS PUBLIC NETWORKS



MAKE ROUTING RULES

To complete the access to restricted sites via openvpn, the following rules need to be added in the route table of the openvpn server:

<cmd>
```
# Allow traffic on the tunnel (tun0) interface.
-A INPUT -i tun0 -j ACCEPT
iptables -A FORWARD -i tun0 -j ACCEPT
iptables -A OUTPUT -o tun0 -j ACCEPT
# Allow forwarding traffic only from the openVPN.
iptables -A FORWARD -i tun0 -o eth0 -s 10.4.0.0/24 -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -t nat -A POSTROUTING -s 10.4.0.0/24 -o eth0 -j MASQUERADE
# Install iptables-persistent for keeping the rules across reboots
sudo apt install iptables-persistent
dpkg-reconfigure iptables-persistent
# Forward IPv4 traffic
echo 'net.ipv4.ip_forward=1' >> /etc/sysctl.d/99-sysctl.conf
sysctl -p # activates the above change
systemctl restart openvpn-server@server.service
```
</cmd>

Then we replace in the client.conf the "client" directive with "tls-client". And we restart openvpn.

If we cannot access the remote resource via openvpn, we can check what's going on with traceroute:

<cmd> traceroute -i tun0 youtube.com </cmd>

While the above command is running, we try to access youtube again from the web browser. traceroute shows us where the connection stops. It might be server or client side error, so we go back to the previous troubleshoots respectively.

CHEAT SHEET

Installation guides:
openvpn.net/community-resources/installing-openvpn/
community.openvpn.net/openvpn/wiki/HOWTO#InstallingOpenVPN

see also https://openvpn.net/community-downloads/
and FAQ https://openvpn.net/community-resources/

Revoke certificates:
openvpn.net/community-=resources/revoking-certificates/

Run openvpn as a service:
https://www.smarthomebeginner.com/configure-openvpn-to-autostart-linux/

julia evan's relevant zine on network troubleshooting:
https://jvns.ca/tcpdump-zine.pdf

how to redirect all web traffic via openvpn:
https://openvpn.net/community-resources/how-to/#redirect

how to run a debian server solely as a openvpn service for all web traffic, includes necessary ip rules:
www.linode.com/docs/networking/vpn/tunnel-your-internet-traffic-through-an-openvpn-server/

Introduction to what a VPN is and how it works:
https://archive.org/details/vpn-zines