OpenVPN - How to Set Up a Slackware Server and a Slackware Client

1. Introduction

1.1. OpenVPN(1)

OpenVPN is an open source software application that implements virtual private network (VPN) techniques for creating secure point-to-point or site-to-site connections in routed or bridged configurations and remote access facilities. It uses a custom security protocol that utilizes SSL/TLS for key exchange. It is capable of traversing network address translators (NATs) and firewalls. It was written by James Yonan and is published under the GNU General Public License (GPL).

OpenVPN allows peers to authenticate each other using a pre-shared secret key, certificates, or username/password. When used in a multiclient-server configuration, it allows the server to release an authentication certificate for every client, using signature and Certificate authority. It uses the OpenSSL encryption library extensively, as well as the SSLv3/TLSv1 protocol, and contains many security and control features.

2. Scope and Objective

The objective of this article is to serve as a tutorial for the readers to set up a basic but functional Slackware VPN Server and Client over the Internet.

The emphasis is to provide a reliable method that can be easily followed to set-up OpenVPN on Slackware Servers and Clients. Nevertheless the process is still not free from pitfalls and require some attention and determination.

This article comprises of a selection of other similar tutorials found on the Internet particularly (2) and (3) and the documents contained in the downloaded source files. However these are reformatted to satisfy the objective.

3. Installation

OpenVPN is already installed on Slackware if a default installation was followed. If this was not the case, then the package is available from the “n” directory of the Slackware repository. Refer to other Slackware specific documents on how to go about this installation.

If you want to confirm that OpenVPN is indeed installed, you can check it by listing the `/var/lib/pkgtools/packages/` directory:

```
# ls /var/lib/pkgtools/packages/openvpn*
```

4. Requirements

Server and a Client computers would be needed. They would have to be connected to the Internet on two different Routers and different Network Routes. For the purpose of this tutorial, specific details are defined in order to enhance the readability. Of course, you will probably have different addresses, so you will need to amend accordingly.

4.1. Server DNS

A URL is normally used to address the Server. This is not mandatory and instead you may use only the Internet IP. However it is recommended to use a URL to access the Server from the Internet, especially if it is connected to a dynamic IP, which is typical for domestic Internet connections. The author is using duckdns.org (4) as it is free upon subscription.

4.2. Server details

```
hostname: server1
IP: 192.168.200.195/255.255.255.0
URL:  servervpn.duckdns.org
Network Interface: eth0
```

4.3. Client details

```
hostname: client1
IP: 192.168.1.101/255.255.255.0
Network Interface: wlan0
```

4.4 Administrator Rights

You will need to have administrator rights to set up OpenVPN. This applies to both the Server and the Client. For simplicity, in this tutorial, it will be assumed that all actions will be performed by the root user. Naturally, advanced users might be more discerning.

4.5 Possible Constraints and Possible Solutions for a WiFi equipped Client

The availability of two Routers might be challenging. Consider that interactive sessions on both the Server and Client will be needed before the VPN is set up. If the Client is equipped with a WiFi interface there might be some easy solutions that may be considered:

1. Use the data smart phone's “Portable Wi-Fi Hot Spot” facility to connect the Client as the VPN Client.
2. Nowadays, many public premises, such as libraries, gardens and Local Councils provide free WiFi service. Other places such as fast food outlets, pubs, cafÃ©s, etc. also provide free WiFi

from their location to their esteemed customers.

5. Creating a Public Key Infrastructure (PKI) using the easy-rsa Scripts

The PKI may be created on any computer, but it is probably more sensible to be done on both the Server and the Client as both would need it. An easy way to build the PKI is to use the `easy-rsa` scripts. These may be downloaded like this:

```
# cd
# git clone http://github.com/OpenVPN/easy-rsa
```

and then archive it for future purposes:

```
# tar czvf easy-rsa.tgz easy-rsa
```

5.1 Create the keys and certificates for the Server

Follow these steps on the Server to create the needed keys and certificates:

```
# cd easy-rsa/easyrsa3
```

Create the PKI and the CA:

```
# ./easyrsa init-pki
# ./easyrsa build-ca
```

Enter a PEM pass phrase, reverify it and then enter a name for the server. In this article I am using the hostnames for clarity (in this case: server1), but you may choose any name.

Then generate the request:

```
# ./easyrsa gen-req server1
```

You will be prompted for the PEM pass phrase, to reverify it and to confirm that the name of the entity is indeed server1. Now you may proceed to sign this request:

```
# ./easyrsa sign-req server server1
```

Confirm the request by entering “yes”, then enter the original ca PEM passphrase.

Now create two additional key files:

```
# cd /etc/openvpn/certs/
# openssl dhparam -out dh2048.pem 2048
# cd /etc/openvpn/keys/
```

```
# /usr/sbin/openvpn --genkey secret ta.key
```

5.2 Create the keys and certificates for the Client

Follow these steps on the Client to create the needed keys and certificates:

You will need the `easy-rsa` scripts, so you can copy the `easy-rsa` tarball from the Server to the Client and extract it:

```
# cd
# tar xvf easy-rsa.tgz
```

Now create the PKI and generate the request:

```
# cd easy-rsa/easyrsa3
# ./easyrsa init-pki
# ./easyrsa gen-req client1
```

You will be prompted for a PEM pass phrase, to re-verify it and to confirm that the name of the entity is indeed client1. In this article I am using the hostnames for clarity (in this case: client1), but you may choose any name.

Copy `pki/reqs/client1.req` back to the Server.

5.2.1 Sign the Client's request on the Server

For the purpose of this article, it is assumed that the Client's request file (`client1.req`) has been transferred to the `$HOME/openvpn/` directory of the Server. Now you can proceed to import and sign the client1 request:

```
# cd $HOME/easy-rsa/easyrsa3
# ./easyrsa import-req $HOME/openvpn/client1.req client1
# ./easyrsa sign-req client client1
```

When prompted enter “yes” and the server1 CA PEM pass phrase.

Copy the generated `$HOME/easy-rsa/easyrsa3/pki/issued/client1.crt` back to the client.

6. Setting up the Server

Copy the following files generated by the easy-rsa scripts to their respective directories in the `/etc/openvpn/` directory:

```
# cp $HOME/easy-rsa/easyrsa3/pki/ca.crt \
> /etc/openvpn/certs/
# cp $HOME/easy-rsa/easyrsa3/pki/issued/server1.crt \
```

```
> /etc/openvpn/certs/
# cp $HOME/easy-rsa/easyrsa3/pki/private/server1.key \
> /etc/openvpn/keys/
```

Copy the provided `server.conf` from the OpenVPN package to the configuration directory:

```
# cp /etc/openvpn/sample-config-files/server.conf /etc/openvpn/
```

Edit the following lines of `/etc/openvpn/server.conf`

From these lines:

```
ca ca.crt
cert server.crt
key server.key  # This file should be kept secret

dh dh1024.pem

;topology subnet

tls-auth ta.key 0 # This file is secret

cipher AES-256-CBC

;log-append  openvpn.log
```

To:

```
ca /etc/openvpn/certs/ca.crt
cert /etc/openvpn/certs/server1.crt
key /etc/openvpn/keys/server1.key #This file should be kept secret

dh /etc/openvpn/certs/dh2048.pem

topology subnet

tls-auth /etc/openvpn/keys/ta.key 0 # This file is secret

data-ciphers-fallback AES-256-CBC

log-append  /var/log/openvpn.log
```

Note that comments in server.conf may start with either start with # or ; In order to help you with entering parameters, the former are used to comment out text while the latter are for commented out configuration lines.

Create a file containing your PEM pass phrase in a secure location; e.g. `/root/password.ovpn` which contains only this pass phrase. Then restrict its permission:

```
# chmod 600 /root/password.ovpn
```

On the Server, edit `/etc/openvpn/server.conf` with the following lines:

```
askpass /root/password.ovpn
auth-nocache
```

Create a directory to store the OpenVPN service PID and restrict its permissions:

```
mkdir /run/openvpn/
chmod 700 /run/openvpn/
```

Give the OpenVPN `rc` script executable permissions, so that the OpenVPN service is started everytime you boot up:

```
# chmod +x /etc/rc.d/rc.openvpn
```

7. Port Forwarding

You will need to forward traffic from the port you have chosen for OpenVPN to be routed to the Server. To accomplish this you will need to provide your Server with a fixed IP and you will need to configure your router. You may use netconfig to set the fixed IP on Slackware. Then you also need to consult the documentation provided with your router to set up the selected IP address reserved for the Server, and the port forwarding. For this default OpenVPN set up, the UDP Port would be 1194.

In case if you have misplaced such documentation, you may search on the Internet on how this may be achieved. A good place to start is https://portforward.com/.

8. Setting up the Client

On the Client machine perform the following instructions to set it up.

Copy the provided `client.conf` from the OpenVPN package to the configuration directory:

```
# cp /etc/openvpn/sample-config-files/client.conf /etc/openvpn/
```

Edit the following lines of /etc/openvpn/client.conf

```
remote my-server-1 1194

;user nobody
;group nobody

ca ca.crt
cert client.crt
key client.key
```

```
tls-auth ta.key 1

cipher AES-256-CBC
```

to the following lines:

```
remote servervpn.duckdns.org 1194

user nobody
group nobody

ca /etc/openvpn/certs/ca.crt
cert /etc/openvpn/certs/client1.crt
key /etc/openvpn/keys/client1.key

tls-auth /etc/openvpn/keys/ta.key 1

data-ciphers-fallback AES-256-CBC

auth-nocache

log-append  /var/log/openvpn.log
```

Note that comments in client.conf may start with either # or ; The former are used to comment out text while the latter are for commented out configuration lines. This should help you a lot in the configuration process.

You will need this file that was generated by the Client's `easy-rsa` scripts:

```
cp $HOME/easy-rsa/easyrsa3/pki/private/client1.key \
> /etc/openvpn/keys/
```

and the following from the Server's `easy-rsa` scripts:

```
$HOME/easy-rsa/easyrsa3/pki/ca.crt
$HOME/easy-rsa/easyrsa3/pki/issued/client1.crt
```

and this file from the Server as well:

```
/etc/openvpn/keys/ta.key
```

Place these files as indicated in `client.conf`. So `ca.crt` and `client1.crt` go under `/etc/openvpn/certs/` while `client1.key` and `ta.key` go under `/etc/openvpn/keys/`

9. Testing the VPN

On the Server:

```
# /etc/rc.d/rc.openvpn restart
```

Enter the Server PEM pass phrase when prompted.

On the Client:

```
# /usr/sbin/openvpn --config /etc/openvpn/client.conf
```

Enter the Client PEM pass phrase when prompted. To stop OpenVPN on the Client just hit CTRL+C

On both you should see a new network interface called tun0. To verify, run this ip command:

```
# /usr/sbin/ip addr show tun0
```

Naturally you can ping the Server from Client (or vice versa):

For example, from the Client:

```
# ping -c 3 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data.
64 bytes from 10.8.0.1: icmp_req=1 ttl=64 time=2888 ms
64 bytes from 10.8.0.1: icmp_req=2 ttl=64 time=1997 ms
64 bytes from 10.8.0.1: icmp_req=3 ttl=64 time=1324 ms

--- 10.8.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 1324.475/2070.293/2888.429/640.527 ms, pipe 3
```

If you fail to set-up a VPN connection, you may want to look in `/var/log/openvpn.log`

```
tal -f /var/log/openvpn.log
```

10. Logrotate (6)

This is logrotate configuration for OpenVPN. It prevents you from ending up with huge OpenVPN log files. `copytruncate` option here is very important - OpenVPN does not want to close the logfile it's writing to, so this file is automatically truncated after copying its contents. Edit this file as `/etc/logrotate.d/openvpn`

```
/var/log/openvpn.log {
    daily
    rotate 12
    compress
    copytruncate
    delaycompress
    missingok
    notifempty
}
```

11. Storing the PEM pass phrase in a secure file and automatic start of service after booting

As hinted in Chapter 6, to start the OpenVPN service on boot, an entry in `/etc/rc.d/rc.local` is not needed as the provided `/etc/rc.d/rc.openvpn` is started by `/etc/rc.d/rc.inet2` at boot up.

As `/etc/rc.d/rc.openvpn` starts OpenVPN with –daemon option, it would not start if you still need to enter the password. In Chapter 6, we showed how this can be resolved for the Server. Naturally, if this is your intention, you may repeat the process for the Client:

On the Client, edit `/etc/openvpn/client.conf` with the following lines:

```
askpass /root/password.ovpn
auth-nocache
```

Create a directory to store the OpenVPN service PID and restrict its permissions:

```
mkdir /run/openvpn/
chmod 700 /run/openvpn/
```

An alternate method (albeit less secure) is to remove the passphrase from `server1.key` and/or `client1.key` files altogether. Ensure to restrict the permissions of the ensuing key. For the Server:

```
# cd /etc/openvpn/keys
# openssl rsa -in server1.key -out tmp.key
```

Enter the pass phrase.

```
# mv tmp.key server1.key
# chmod 600 server1.key
```

If you had them remove these lines from `/etc/openvpn/server.conf`

```
askpass /root/password.ovpn
auth-nocache
```

Similarly, this can be repeated for the Client:

```
# cd /etc/openvpn/keys
# openssl rsa -in client1.key -out tmp.key
```

Enter the pass phrase.

```
# mv tmp.key client1.key
# chmod 600 client1.key
```

Then, if you had it, remove this line from `/etc/openvpn/client.conf`

```
askpass /root/password.ovpn
```

If you intend to use the `rc` script on the Client, proceed like this:

```
mkdir /run/openvpn/
chmod 700 /run/openvpn/
chmod +x /etc/rc.d/rc.openvpn
```

12. IP Routing

Up to now we have created a tunnel device on both the Server and the Client called tun0 which is visible only to these two machines. However more work is needed to route the Client's connection via tun0 and then to the WAN that is connected to the Server.

12.1 IP Forwarding

On the Server, enable IP forwarding:

```
# chmod +x /etc/rc.d/rc.ip_forward
# /etc/rc.d/rc.ip_forward start
```

IP forwarding is now enabled and will be enabled also after you reboot.

Make a directory called ccd in /etc/openvpn

```
# mkdir /etc/openvpn/ccd/
```

Create a file with the same name of the client (in this case `client1`) and enter the following line in `/etc/openvpn/ccd/client1`

```
iroute 192.168.1.0 255.255.255.0
```

Replace 192.168.1.0 255.255.255.0 by the Network Route of your Client.

Similarly edit `/etc/openvpn/server.conf` with the following lines:

```
push "route 192.168.200.0 255.255.255.0"

client-config-dir /etc/openvpn/ccd
route 192.168.1.0 255.255.255.0

push "redirect-gateway def1 bypass-dhcp"

push "dhcp-option DNS 208.67.222.222"
push "dhcp-option DNS 208.67.220.220"
```

Naturally replace 192.168.200.0 255.255.255.0 with the Server's Network Route, and 192.168.1.0

255.255.255.0 with the Client's Network Route. 208.67.222.222 and 208.67.220.220 are the OpenDNS IPv4 DNS service addresses.


Up to now the DNS push configuration has not been successful.

You can either use the original Client DNS servers or else you may rewrite /etc/resolv.conf manually:

```
# OpenDNS Servers
nameserver 208.67.222.222
nameserver 208.67.220.220
```

According to your routing table however, it is still worth trying to use the DNS servers listed by the Client, I find that they are generally still available, so you would not need to do anything. However do be aware of possible DNS leaks if you are concerned about your privacy.

Some users have reported that their Client's Network Manager, (or any other similar application) re-wrote the original /etc/resolv.conf back after their manual editing. This could not be reproduced by the author of this article (yet), but you may consider installing and configuring openresolv(5) if this actually happens to you. A SlackBuild for openresolv may be found on http://slackbuilds.org. Openresolv is currently out of the scope of this article.

Next you will have to configure NAT forwarding on the Server (only).

You can do this by means of iptables or the newer nftables (not both).

12.2 NAT forwarding with iptables

Start by flushing the `iptables`

```
# /usr/sbin/iptables -F
```

And then:

```
# /usr/sbin/iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j
MASQUERADE
```

On Slackware, such a line may be included in `/etc/rc.d/rc.firewall` and `/etc/rc.d/rc.inet2` will run it each time you reboot the Server if the former has executable permissions. You do not have to include anything in `/etc/rc.d/rc.local`.

The exact lines which you need to include depend on whether you already entered your own iptables filter chains and rules, but I will assume that that this is not the case.

As already explained, as a minimum you only need to enter the following lines in `/etc/rc.d/rc.firewall`

```
#!/bin/sh
iptables -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE
```

Give the firewall `rc` script executable permission:

```
# chmod +x /etc/rc.d/rc.firewall
```

and start it:

```
# /etc/rc.d/rc.firewall start
```

Restart the OpenVPN service on the Server:

```
# /etc/rc.d/rc.openvpn restart
```

and reconnect from the Client:

```
# /usr/sbin/openvpn /etc/openvpn/client.conf
```

12.3 NAT forwarding with nftables

If you prefer `nftables`, these are the commands you can use:

```
/usr/sbin/nft flush ruleset
/usr/sbin/nft add table nat
/usr/sbin/nft 'add chain nat postrouting { type nat hook postrouting
priority 100 ; }'
/usr/sbin/nft add rule nat postrouting ip saddr 10.8.0.0/24 oif eth0
masquerade
```

13. Firewalls

You may find that on some networks, UDP port 1194 is blocked, and so the Client will be unable to connect. In order to penetrate through the firewall you may want to try changing the port to 443 - normally reserved for https. Using TCP instead of UDP will also help. To make these change you will need to amend /etc/openvpn/server.conf of the Server, from

```
port 1194
proto udp
```

to:

```
port 443
proto tcp
```

Also, comment out `explicit-exit-notify 1` in `/etc/openvpn/server.conf`

You also have to modify your Router's port forwarding to TCP port 443.

Edit `/etc/openvpn/client.conf` of the Client, from

```
proto udp

remote servervpn.no-ip.org 1194
```

to:

```
proto tcp

remote servervpn.no-ip.org 443
```

14. Sources

(1) http://en.wikipedia.org/wiki/OpenVPN

(2) https://wiki.archlinux.org/index.php/OpenVPN

(3) http://slackwiki.com/OpenVPN_smcr_2012

(4) http://www.duckdns.org

(5) http://roy.marples.name/projects/openresolv/index

(6) https://gist.github.com/adaRn/337838d87cd8f544407d6446cf7daa80

- Written for Slackware 15.0 in April 2022
- Originally written by Chris Abela

howtos, network, openvpn