# 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 DVD. 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/log/packages/ directory:

```
# ls /var/log/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 a 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 noip2(4) as it is free upon subscription. A noip2 slackbuild is available from http://slackbuilds.org.

## 4.2. Server details

```
hostname: server1
IP: 192.168.200.195/255.255.255.0
URL:  servervpn.no-ip.org
Network Interface: eth0
```

## 4.3. Client details

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

## 4.3 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.

## 4.4 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 a 3G smart phone's “Portable Wi-Fi Hot Spot” facility to connect the Client as the VPN Client. As 3G bandwidth is expensive you may want to minimise traffic. For example, you might want to switch off services that are not absolutely essential during the course of this exercise,

such as ntpd, dropbox and tor.
2. Connect the Client to another WiFi available in the vicinity of the Server. Some lucky people live in areas were benevolent neighbours provide them with openly accessible Internet WiFi. It is recommended to request permission before taking up this solution. In case that no such open service exists, you may find it appropriate to request a temporary password from a friendly neighbour for the private encrypted WiFi service.
3. Nowadays, many governmental premises, such as libraries and Local Councils provide free WiFi service. Other places such as fast food outlets, pubs, cafés, etc. also provide free WiFi from their location to their valuable customers. You may access the Server via an available service such as SSH from a WiFi equipped Client. If this option is chosen for this solution, be aware that the Client may have to pass through some firewalls. Besides the VPN connection might be a breach of the terms of conditions that should be accepted before using the WiFi service.

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

The PKI may be created on any computer with a VPN installation, 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 cvf easy-rsa.tar 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 another 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.tar
```

Now create the PKI and generate the request:

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

You will be prompted for another 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 sample server.conf from the OpenVPN source onto the OpenVPN's configuration directory. The source of OpenVPN may be obtained from Slackware's source DVD or your favourite Slackware mirror or from http://openvpn.net. In the following example, I am downloading the source from ftp.slackware.com

```
# cd /tmp/
# wget -c \
>
ftp://ftp.slackware.com/pub/slackware/slackware/source/n/openvpn/openvpn-*.t
ar.?z
# cd /usr/src/
# tar xvf /tmp/openvpn-*.tar.?z
```

Copy the file server.conf contained in the source to the OpenVPN configuration directory:

```
# cp openvpn-*/sample/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
;tls-auth ta.key 0 # This file is secret

;user nobody
;group nobody

;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

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

user nobody
group nobody

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

Finally add the following to /etc/openvpn/server.conf:

```
# Select a cryptographic cipher.
# This config item must be copied to
# the client config file as well.
cipher AES-256-CBC
# If you want to use OpenVPN as a daemon, uncomment this line.
# Generally speaking, servers should run OpenVPN as a daemon
;daemon
```

An attentive reader may be tempted to uncomment the 'daemon' option. This would not allow the user to enter the pass phrase, therefore it would not work until the pass phrase is defined in server.conf as described in Chapter 10.

```
# cat /var/log/openvpn.log
```

Note that comments in server.conf may be 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.

Copy the rc.openvpn listed hereunder and place under /etc/rc.d/

```
#!/bin/sh
#
# /etc/rc.d/rc.openvpn
#
# Start/stop/restart the OpenVPN server.
#

ovpn_start() {
  if [ -x /usr/sbin/openvpn -a -r /etc/openvpn/server.conf ]; then
    echo "Starting OpenVPN:  /usr/sbin/openvpn server.conf"
    /usr/sbin/openvpn /etc/openvpn/server.conf
  fi
}

ovpn_stop() {
  killall openvpn
```

```
}

ovpn_restart() {
  ovpn_stop
  sleep 2
  ovpn_start
}

case "$1" in
'start')
  ovpn_start
  ;;
'stop')
  ovpn_stop
  ;;
'restart')
  ovpn_restart
  ;;
*)
  echo "Usage: $0 {start|stop|restart}"
esac
```

Then give it executable permissions:

```
# chmod 755 /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, network-manager or wicd 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 our 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 http://portforward.com/.

# 8. Setting up the Client

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

Download the OpenVPN source tarball and extracted it as explained in Chapter 6, then proceed to copy the included configuration file for clients:

```
# cp /usr/src/openvpn-*/sample/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
```

to the following lines:

```
remote servervpn.no-ip.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
```

Finally add the following to /etc/openvpn/client.conf:

```
# Select a cryptographic cipher.
# This config item must be copied to
# the server config file as well.
cipher AES-256-CBC
```

Note that comments in client.conf may be 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 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 start
```

Enter the Server PEM pass phrase when prompted.

On the Client:

```
# /usr/sbin/openvpn /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. On the Server, I obtained the following:

```
# ifconfig tun0
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.8.0.1  netmask 255.255.255.255  destination 10.8.0.2
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen
100  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Similarly on the Client:

```
# ifconfig tun0
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 10.8.0.6  netmask 255.255.255.255  destination 10.8.0.5
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen
100  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

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
```

# 10. Storing the PEM pass phrase in a secure file and Automatic start of service after booting

To start the OpenVPN service on boot, an entry in /etc/rc.d/rc.local is needed, but you would have to enter the server PEM pass phrase every time. This might be undesirable if the Server is unreachable. If this is the case, 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
```

Also, uncomment the 'daemon' option.

This may be repeated also on the Client, just edit /etc/openvpn/client.conf instead of /etc/openvpn/server.conf.

To start the OpenVPN service automatically on boot-up from the Server, include these lines in /etc/rc.d/rc.local

```
# Start the OpenVPN Service
if [ -x /etc/rc.d/rc.openvpn ]; then
  /etc/rc.d/rc.openvpn start
fi
```

An alternate method (albeit less secure) is to remove the passphrase from server1.key file altogether. Don't forget to set permissions on the key to avoid it being world-readable.

```
# cd /etc/openvpn/keys
# openssl rsa -in server1.key -out tmp.key
# mv tmp.key server1.key
# chmod 600 server1.key
```

# 11. 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.

## 11.1 Server Configuration

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 IP 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 some iptables NAT forwarding on the Server (only). You can do this by first flushing the iptables:

```
# iptables -F
```

And then:

```
# 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
```

If on the other hand you would like a better firewall and you are at least moderately confident with iptables, I propose the following script to be included in your /etc/rc.d/rc.firewall. The comments in the script should help you understand the impact they will have on the Server.

```
#!/bin/bash
# Start/stop/restart/status the firewall
IPT=/usr/sbin/iptables # This will provide some portability
firewall_start() {
  # flush the iptables
  echo -e "Starting the firewall ....\c"
  $IPT -F
  # policies
  $IPT -P OUTPUT DROP
  $IPT -P INPUT DROP
  $IPT -P FORWARD DROP

  $IPT -N SERVICES # services is a custom chain

  # allowed output
  $IPT -A OUTPUT -o lo -j ACCEPT
  $IPT -A OUTPUT -o eth0 -j ACCEPT
  $IPT -A OUTPUT -o tun0 -j ACCEPT

  # allowed inputs
  #$IPT -A INPUT -i lo -j ACCEPT # uncomment if the host is a desktop
  $IPT -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT # allow
```

```
responses
  $IPT -A INPUT -j SERVICES # append the services chain to the input

  # allowed forwarding for OpenVPN
  $IPT -A FORWARD -i eth0 -o tun0 -m state --state ESTABLISHED,RELATED -j
ACCEPT
  $IPT -A FORWARD -s 10.8.0.0/24 -o eth0 -j ACCEPT

  # masquerade the OpenVPN network
  $IPT -t nat -A POSTROUTING -s 10.8.0.0/24 -o eth0 -j MASQUERADE

  # allow sshd on the default tcp port 22
  #$IPT -A SERVICES -p tcp --dport 22 -j ACCEPT # Uncomment to allow sshd

  # allow OpenVPN for the default udp port 1194
  $IPT -A SERVICES -p udp --dport 1194 -j ACCEPT
  echo "done."
}

firewall_stop() {
  echo -e "Stopping the firewall ....\c"

  # polcies (permissive)
  $IPT -P OUTPUT ACCEPT
  $IPT -P INPUT ACCEPT
  $IPT -P FORWARD ACCEPT

  # flush the iptables
  $IPT -F

  # delete the services custom chain
  $IPT -X SERVICES
  echo "done."
}

firewall_status() {
  $IPT -vL
}

case "$1" in
'start')
  firewall_start
  ;;
'stop')
  firewall_stop
  ;;
'restart')
  firewall_stop
  firewall_start
  ;;
'status')
```

```
  firewall_status
  ;;
*)
  echo "Usage $0 start|stop|restart|status"
esac
```

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. Firewalls

In the previous chapter we referred to a firewall you may include to protect your OpenVPN Server. However this chapter refers to firewalls on the Client LAN that may block the VPN connection by blocking traffic on UDP port 1194.

In order to penetrate through the Client firewall your 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
```

and /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
```

The Server's firewall script would also need to be modified. Change these lines:

```
# allow vpn on the default udp port 1194
$IPT -A SERVICES -p udp --dport 1194 -j ACCEPT
```

to:

```
# allow vpn on the custom tcp port 443
$IPT -A SERVICES -p tcp --dport 443 -j ACCEPT
```

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

## 13. 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.no-ip.com

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

- Written for Slackware 14.2 in April 2018
- Originally written by Chris Abela

howtos, network, openvpn